AGRORENT HUB

Mini Project Report

Submitted by

Wilgimol Thomas

Reg. No.: AJC20MCA-I057

In Partial fulfillment for the Award of the Degree Of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2024-2025

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, **"AGRORENT HUB"** is the bona fide work of **WILGIMOL THOMAS (Regno: AJC20MCA-I057)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2024-25.

Mr. G. S. Ajith — **Mr. Binumon Joseph**

Internal Guide — **Coordinator**

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report **"AGRORENT HUB"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the **Integrated Master of Computer Applications (INMCA)** from **APJ Abdul Kalam Technological University**, during the academic year **2024-2025**.

Date: **WILGIMOL THOMAS**

KANJIRAPPALLY **Reg: AJC20MCA-I057**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Director (Administrator) **Rev. Fr. Roy Abraham Pazhayaparambil** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Mr. Binumon Joseph** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr. G. S. Ajith** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

WILGIMOL THOMAS

ABSTRACT

.

AgroRent Hub is an online platform built with Django to facilitate the rental of agricultural products, equipment, workers, and services. It supports multiple user roles: Farmers (Customers), Suppliers, Delivery Boys, and an Admin. Suppliers can list items for rent, while Farmers can browse, add items to their cart, and rent them for a specified period. Delivery Boys handle the delivery and return of rented items, covering repair costs if defects are found.

Users register by selecting their role. The Admin manages all users and transactions. Suppliers have dashboards to manage inventory and stock levels. Farmers can view products, manage rentals, and make payments. Delivery Boys manage delivery assignments and update statuses.

Key features include OTP-based verification, secure payment processing, and low inventory alerts. Users receive notifications for orders and returns. Django ensures scalability, security, and efficient data management, with RESTful APIs enabling smooth frontend-backend communication. The platform is user-friendly and secure.

CONTENT

List of Abbreviation

1. AgroRent Hub - ARH
2. Application Programming Interface - API
3. Content Management System - CMS
4. Customer Relationship Management - CRM
5. Data Management System - DMS
6. Electronic Data Interchange - EDI
7. Geographic Information System - GIS
8. Hypertext Markup Language - HTML
9. JavaScript Object Notation - JSON
10. Machine Learning - ML
11. Payment Gateway - PG
12. Product Lifecycle Management - PLM
13. Return on Investment - ROI
14. Secure Socket Layer - SSL
15. User Interface - UI
16. User Experience - UX
17. Virtual Private Network - VPN
18. Web Application - WA
19. Web Development - WD
20. World Wide Web - WWW

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

AgroRent Hub is an online platform built with Django to facilitate the rental of agricultural products, equipment, workers, and services. It supports multiple user roles: Farmers (Customers), Suppliers, Delivery Boys, and an Admin. Suppliers can list items for rent, while Farmers can browse, add items to their cart, and rent them for a specified period. Delivery Boys handle the delivery and return of rented items, covering repair costs if defects are found.

Users register by selecting their role. The Admin manages all users and transactions. Suppliers have dashboards to manage inventory and stock levels. Farmers can view products, manage rentals, and make payments. Delivery Boys manage delivery assignments and update statuses.

Key features include OTP-based verification, secure payment processing, and low inventory alerts. Users receive notifications for orders and returns. Django ensures scalability, security, and efficient data management, with RESTful APIs enabling smooth frontend-backend communication. The platform is user-friendly and secure.

1.2 PROJECT SPECIFICATION

AgroRent Hub is an innovative online platform designed to facilitate the rental of agricultural products, equipment, workers, and services, utilizing the Django framework for secure and scalable development. The platform caters to multiple user roles, including Farmers (Customers), Suppliers, Delivery Boys, and Admins, each with distinct functionalities. Farmers can browse, filter, and rent products, manage their rental durations, and make secure payments while tracking their orders and selecting delivery boys for their rentals. Suppliers have the ability to manage their inventory, handle rental requests, monitor stock levels, and oversee financial transactions, thereby ensuring that popular items are readily available. Delivery Boys are responsible for managing assigned deliveries, updating the status of returns, and reporting any defects related to rented items. The Admin module provides oversight of user accounts and rental transactions, maintaining the overall integrity of the platform.

Key features include OTP-based verification for secure user authentication, low inventory alerts for suppliers, and user notifications regarding order statuses and returns. The implementation of a demand forecasting feature aims to predict future rental demand based on historical data and seasonal trends, helping suppliers optimize their inventory management. The project leverages established technologies, ensuring technical feasibility, and is designed to be user-friendly, promoting quick adoption among all user types. A cost-benefit analysis supports the economic feasibility of the platform, justifying development costs through enhanced operational efficiency and increased customer satisfaction. Overall, AgroRent Hub presents a comprehensive solution for streamlining agricultural rentals, enhancing user experience, and ensuring effective service delivery in the agricultural sector.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The AgroRent Hub is an online platform designed to revolutionize the rental process within the agricultural sector by providing a comprehensive solution for renting agricultural products, equipment, services, and labor. This system study aims to explore the architecture, functionality, and benefits of the AgroRent Hub, focusing on its capacity to streamline operations among various user roles, including Farmers (Customers), Suppliers, Delivery Boys, and Admins.

In an industry where efficiency and accessibility are paramount, AgroRent Hub serves as a centralized marketplace, enabling seamless interactions between stakeholders. Farmers can easily browse, filter, and rent products based on their specific needs, while Suppliers gain robust tools for managing inventory, tracking rentals, and handling financial transactions. Delivery Boys facilitate timely and efficient delivery and return services, ensuring that the rental experience is smooth and reliable.

This system study will delve into the underlying technologies employed in AgroRent Hub, including the Django framework for backend development and HTML, CSS, and JavaScript for frontend design. Additionally, it will examine the key features that enhance user experience, such as secure payment processing, OTP-based verification, and demand forecasting. Through this analysis, the AgroRent Hub aims to demonstrate its potential to significantly improve operational efficiency, reduce costs, and increase overall customer satisfaction in the agricultural rental market.

2.2 LITERATURE REVIEW

Recent advancements in image processing and AI have led to significant strides in agriculture, revolutionizing areas such as crop monitoring, disease detection, and equipment management. Core image processing techniques—such as image enhancement, filtering, and edge detection—lay the foundation by improving image clarity and detail, while more sophisticated methods, including object detection, segmentation, and the use of deep learning models like convolutional neural networks (CNNs), enable nuanced analyses of agricultural imagery [7], [15]. CNNs, in particular, excel in identifying intricate patterns within agricultural datasets, which is essential for the precise classification of crops, farm equipment, and environmental factors. Studies have shown that CNNs can achieve high accuracy in classifying crop types, identifying pests, and diagnosing plant diseases, thereby contributing to more efficient resource use and higher crop yields [1], [16].

The introduction of multispectral and hyperspectral imaging has added a new dimension to agricultural monitoring. These imaging techniques capture data across various light spectra, including wavelengths beyond the visible range, making them ideal for assessing crop health, evaluating soil properties, and detecting early signs of disease or nutrient deficiencies. Such imaging helps farmers make informed decisions regarding irrigation, fertilization, and pest control, thus optimizing productivity and resource conservation [8]. When combined with machine learning algorithms, these techniques enable the development of predictive models that support tasks like crop yield forecasting, soil moisture monitoring, and disease outbreak prediction. This predictive capability is particularly beneficial in precision agriculture, where efficient water usage and sustainable farming practices are priorities [9], [13].

For agricultural rental platforms, image processing offers substantial advantages, particularly in equipment management. Automated image classification systems can categorize and tag equipment images uploaded by suppliers, minimizing the need for manual categorization and reducing errors. By employing CNNs trained to recognize specific types of agricultural equipment—such as tractors, plows, or harvesters—these platforms can automate the classification and organization of inventory, making the rental process smoother and more efficient for both suppliers and users [12], [20]. Research highlights that integrating such models within digital platforms improves user experience by enabling accurate, real-time classification and streamlining inventory management processes [10]. This automation also facilitates faster equipment matching for users, contributing to increased rental platform efficiency.

However, several challenges remain in implementing real-time, accurate image processing on agricultural platforms. Environmental variability—such as changes in lighting, weather conditions, and background noise—can impact image quality and the accuracy of model predictions. To address these issues, building a diverse and comprehensive dataset of equipment images under varied conditions is essential. This approach helps models generalize better, thus improving classification accuracy across different scenarios [15]. Furthermore, real-time image processing requires highly optimized models that can balance

speed and accuracy without overloading the platform's resources. Efficient integration of image processing with other system components, like payment gateways and demand forecasting tools, is also crucial for a seamless user experience.
Looking forward, advancements in deep learning architectures, such as transformers and generative models, are expected to further enhance image analysis in agriculture. Additionally, the integration of IoT (Internet of Things) with image processing can enable real-time monitoring and predictive maintenance of rental equipment, potentially reducing downtime and repair costs. Such innovations, when applied to platforms like AgroRent Hub, could lead to a robust ecosystem where users benefit from quick and accurate access to rental equipment, and suppliers experience streamlined inventory management, secure transactions, and enhanced demand prediction capabilities. This transformation makes agricultural rentals more accessible, reliable, and beneficial for all stakeholders involved [18], [21], [23].

2.3 PROPOSED SYSTEM

The AgroRent Hub proposes a robust, user-friendly platform designed to overcome the limitations of the existing system. It integrates multiple user roles—Farmers, Suppliers, Delivery Boys, and Admins—into a single platform with comprehensive functionalities for each role. The system features secure payment processing, advanced inventory management with demand forecasting, OTP-based user verification, and efficient delivery tracking. The platform also allows suppliers to manage stock levels dynamically, monitor financial transactions, and ensure a smooth rental experience for Farmers. Delivery Boys can easily manage assigned deliveries and report defects, while Admins oversee the entire process and ensure the platform operates efficiently.

2.4 ADVANTAGES OF PROPOSED SYSTEM

- Automation of Manual Tasks: AgroRent Hub automates various tasks like inventory management, order tracking, and payment processing, significantly reducing human error and increasing efficiency.
- Secure Transactions: The platform ensures secure payments through OTP-based verification and encryption, offering a safer user experience.
- User-Friendly Interface: AgroRent Hub is designed with a simple, intuitive interface

to ensure ease of use for all stakeholders, including non-technical users like Farmers and Delivery Boys.

- Comprehensive Role-Based Management: Each user role has specific functionalities, ensuring that each stakeholder has the tools necessary for their tasks, whether it's managing inventory, processing payments, or handling deliveries.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study evaluates the practicality of a proposed system by examining various factors such as costs, technical capabilities, and user behavior. This section addresses the economic, technical, and behavioral feasibility of AgroRent Hub.

3.1.1 Economical Feasibility

The economic feasibility of AgroRent Hub examines whether the financial investment in developing and maintaining the platform is justified by the expected benefits. The development costs include software tools, personnel, and maintenance. With AgroRent Hub, the platform's efficiency in managing rentals, handling payments, and reducing manual errors directly leads to increased transaction volume and reduced administrative overhead, improving profitability. Furthermore, by offering suppliers demand forecasting tools and farmers a seamless rental process, the platform promotes better resource utilization, leading to long-term financial sustainability. The potential revenue streams, such as service fees, premium features, or advertisements, ensure a solid return on investment, making the project economically viable.

3.1.2 Technical Feasibility

AgroRent Hub leverages established technologies like Django for backend development, HTML/CSS/JavaScript for the frontend, and SQLite for database management. These technologies are well-documented, widely supported, and ensure scalability, security, and efficient data handling. The chosen tools have been successfully used in similar projects, providing confidence in the technical stability and ease of maintenance for the platform. Furthermore, with secure payment gateways and OTP verification, the system ensures user data protection and transaction security. The infrastructure can be easily expanded, allowing the platform to scale with an increasing number of users and rentals, ensuring its technical feasibility.

3.1.3 Behavioral Feasibility

The behavioral feasibility assesses how well users—Farmers, Suppliers, Delivery Boys, and Admins—will adopt the platform. AgroRent Hub is designed with an intuitive, user-friendly interface that accommodates users with varying levels of technical expertise. Farmers, who may not be familiar with complex systems, can easily browse and rent products, while suppliers can efficiently manage inventory and financial transactions. The platform's real-time updates, secure payment methods, and notifications ensure smooth user interaction, leading to quick adoption and minimal training requirements. The design of the system emphasizes accessibility and simplicity, ensuring that all users, including non-technical stakeholders, will find the platform effective and easy to use.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	-	Intel Core i5 or higher (or equivalent)
RAM	-	4 GB or above
Hard Disk	-	500 GB HDD or SSD

3.2.2 Software Specification

Front End	-	HTML, CSS, JavaScript, jQuery, AJAX
Backend	-	Django (using SQLite or MySQL for database management)
Client on PC	-	Windows 7 and above
Technologies Used (for machine	-	JavaScript, HTML5, AJAX, jQuery, Django), Bootstrap, Selenium testing), TensorFlow/Keras (for deep learning), Scikit-learn (for learning).

3.3 SOFTWARE DESCRIPTION

3.3.1 Python

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is widely used in web development, data science, machine learning, and automation due to its rich ecosystem of libraries and frameworks.

- Key Features:
    - Simple and easy-to-read syntax.
    - Extensive standard library and third-party modules.
    - Cross-platform compatibility.

    - Ideal for rapid development.
    - Widely used in machine learning, data science, and web development.
- Version Used: Python 3.x (supports Django, TensorFlow, Selenium, and other key technologies used in the project).

3.3.2 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It simplifies the process of building robust, scalable web applications by providing pre-built components and modules, such as authentication, URL routing, and database management. Django follows the Model-View-Template (MVT) architectural pattern and emphasizes reusability, less code, and fast development.

Key Features:

- Secure: Helps developers avoid common security issues like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
- Scalable: Suitable for small to large applications.
- Built-in Admin Interface: Provides an out-of-the-box admin interface to manage database models.
- ORM: Django's Object-Relational Mapping (ORM) simplifies database operations by allowing developers to interact with the database using Python code rather than SQL.
- Modular: Promotes code reusability and follows a modular design.

Version Used: Django 3.x or higher (compatible with Python 3.x and suited for scalable web applications).

Both Python and Django work together seamlessly, making them ideal for building complex platforms like AgroRent Hub, which involves managing multiple user roles, handling payments, and supporting machine learning models.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

System design involves defining the architecture, components, modules, and interfaces of a system to satisfy specified requirements. It plays a critical role in determining the functionality, scalability, and maintainability of the application. The design phase bridges the gap between the system requirements and its implementation.

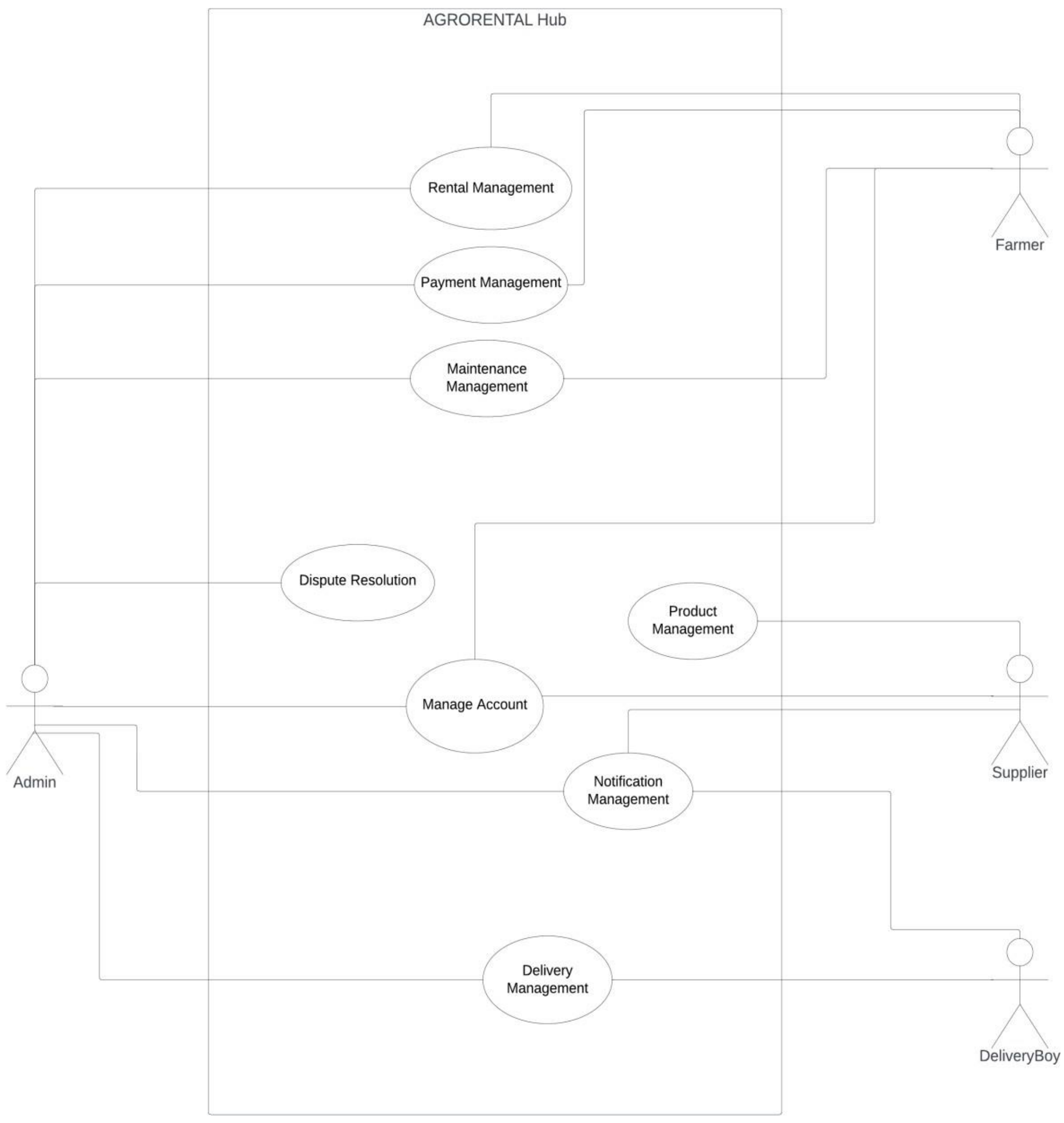
In AgroRent Hub, system design ensures smooth interaction between multiple user roles (Farmers, Suppliers, Delivery Boys, and Admin), efficient handling of rental processes, payments, and the integration of machine learning models. UML diagrams such as Use Case, Sequence, Activity, Class, and Component Diagrams help visualize the architecture and relationships between system components.

4.2 UML DIAGRAM

4.2.1 USE CASE DIAGRAM

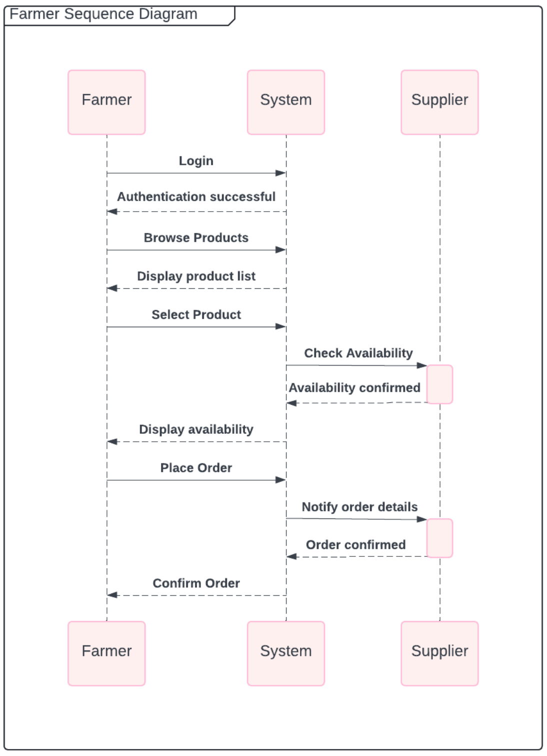
The Use Case Diagram illustrates the interaction between users (actors) and the system. It defines the main functionality provided by the system to its users. In AgroRent Hub, the main actors are Farmers (Customers), Suppliers, Delivery Boys, and Admins.

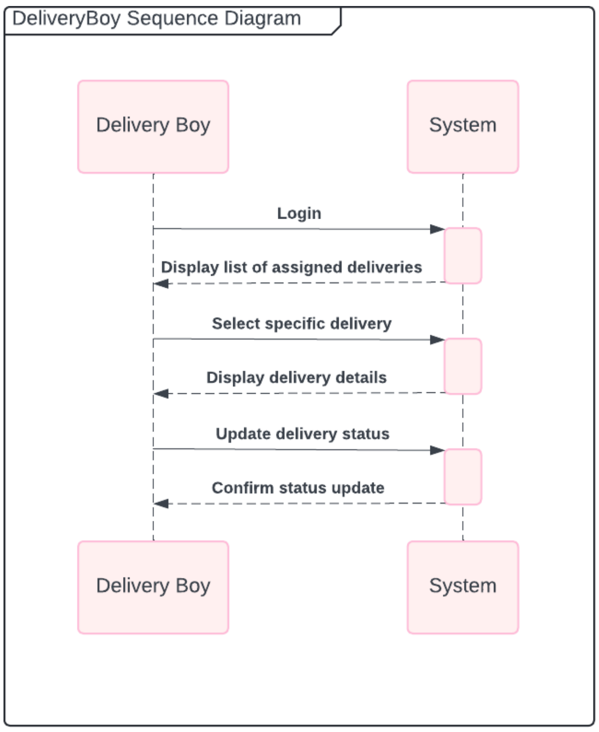
- Actors: Farmers, Suppliers, Delivery Boys, Admin.
- Use Cases:
    - Farmers can browse products, add items to cart, make payments, and track orders.
    - Suppliers can manage their inventory and view rental requests.
    - Delivery Boys update delivery statuses and report defects.
    - Admins monitor users, transactions, and system settings.

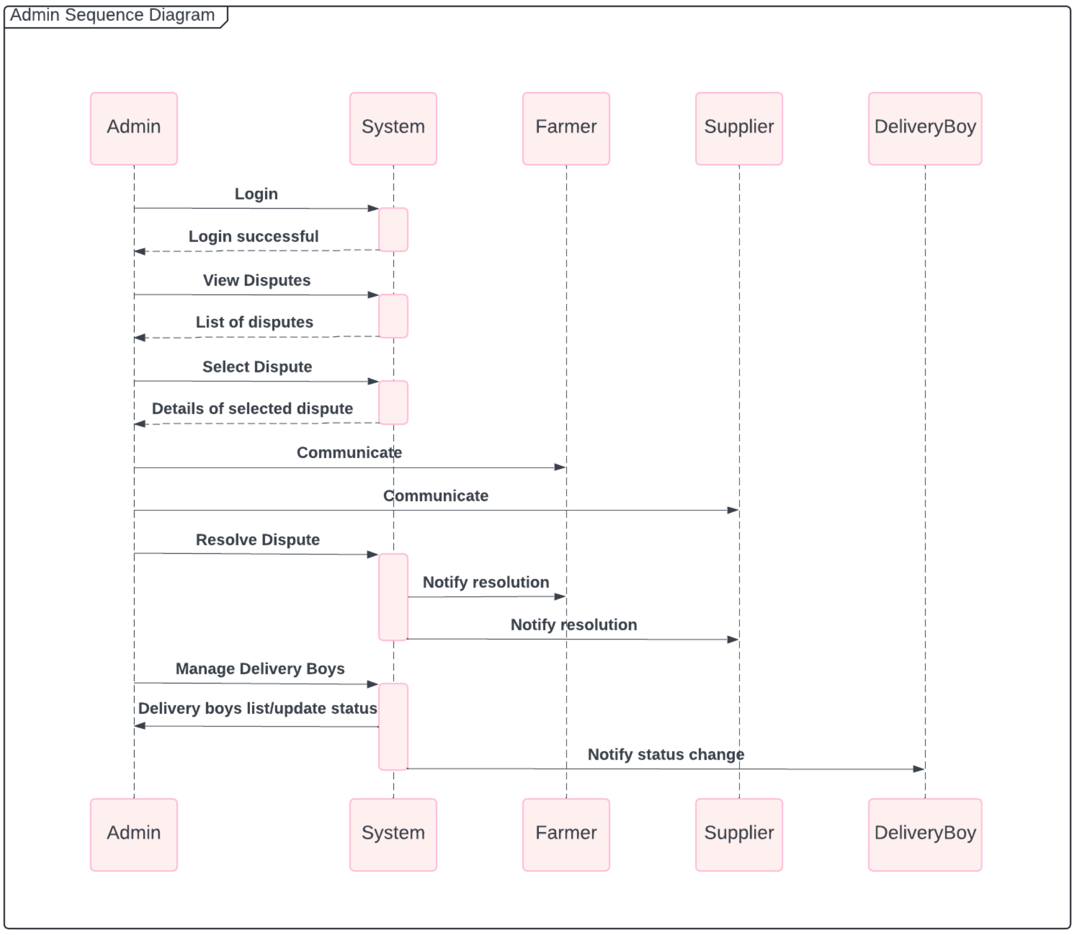


4.2.2 SEQUENCE DIAGRAM

A Sequence Diagram depicts the sequence of interactions between objects or components over time. It shows how different entities in the system interact to complete a particular process.


Farmer Sequence Diagram
Farmer
System
Supplier
Login
Authentication successful
Browse Products
Display product list
Select Product
Check Availability
Availability confirmed
Display availability
Place Order
Notify order details
Order confirmed
Confirm Order
Farmer
System
Supplier

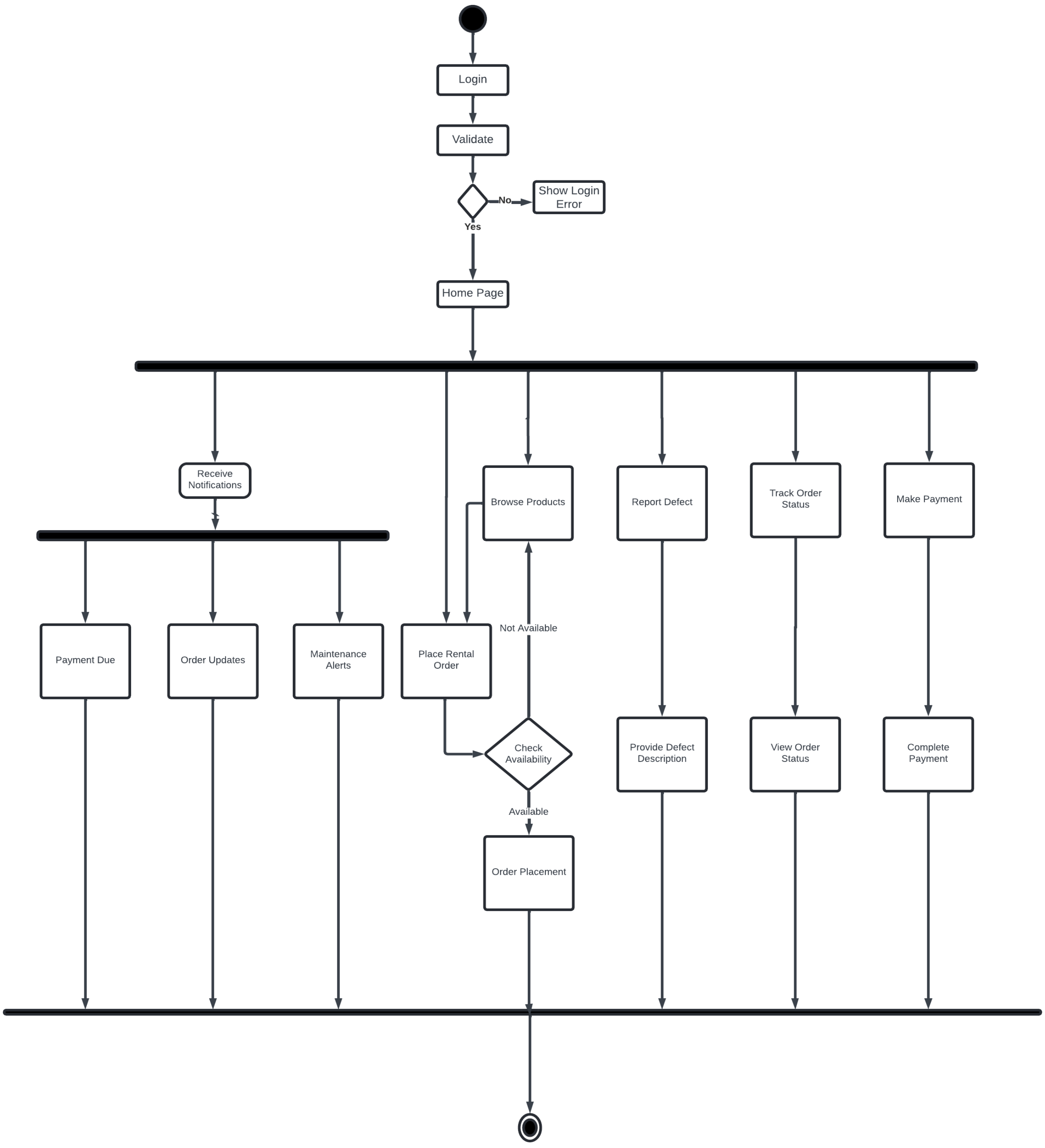

DeliveryBoy Sequence Diagram
Delivery Boy
System
Login
Display list of assigned deliveries
Select specific delivery
Display delivery details
Update delivery status
Confirm status update
Delivery Boy
System

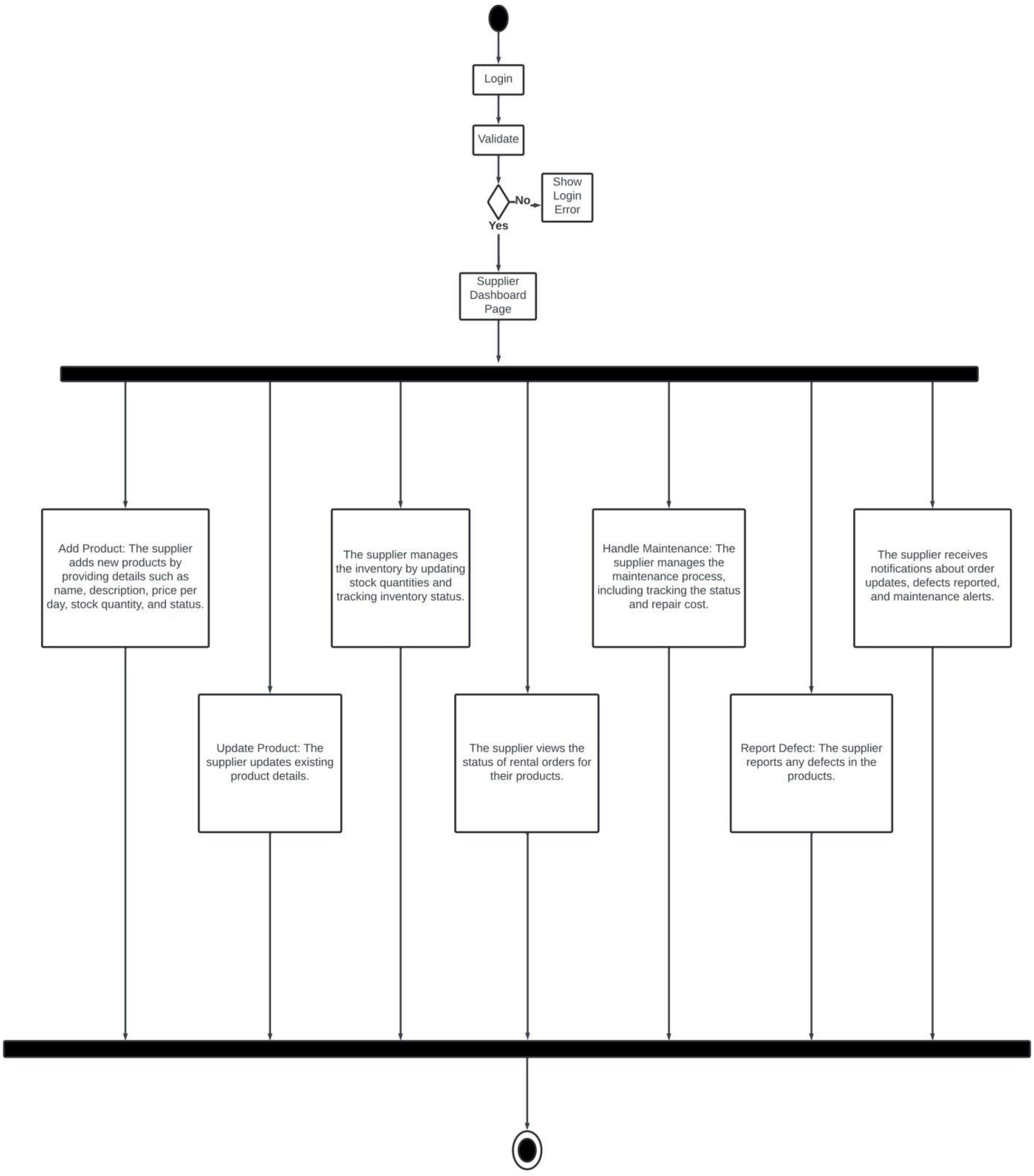

Admin Sequence Diagram
Admin
System
Farmer
Supplier
DeliveryBoy
Login
Login successful
View Disputes
List of disputes
Select Dispute
Details of selected dispute
Communicate
Communicate
Resolve Dispute
Notify resolution
Notify resolution
Manage Delivery Boys
Delivery boys list/update status
Notify status change
Admin
System
Farmer
Supplier
DeliveryBoy

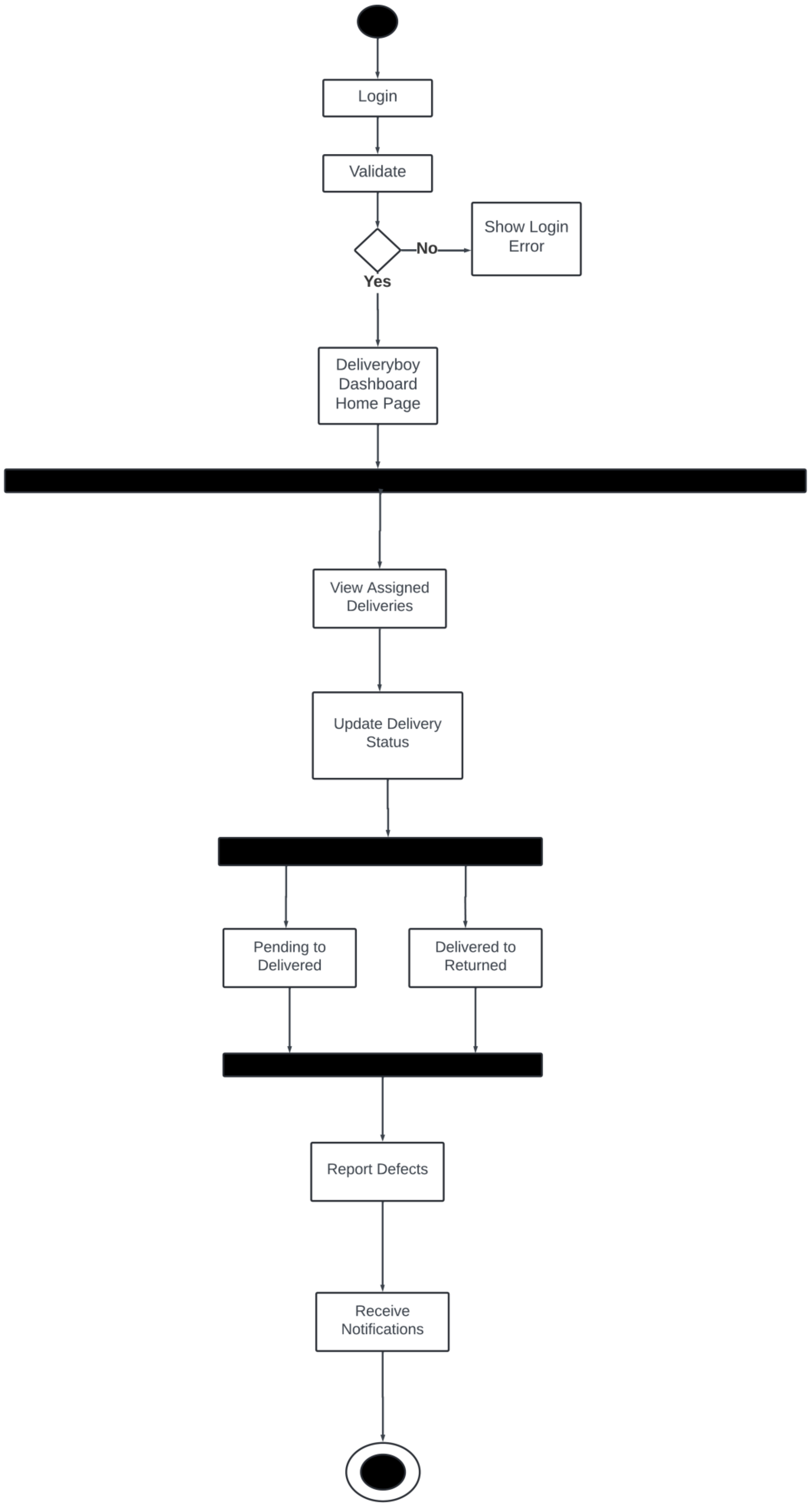
4.2.4 Activity Diagram

An Activity Diagram describes the workflow of activities and actions. It is useful for illustrating how tasks are performed in the system.

- Example: The process of returning a product.
  - Activity starts with the Farmer scheduling a return.
  - Delivery Boy picks up the product and updates the status.
  - Supplier inspects the product and approves the return or applies repair costs.



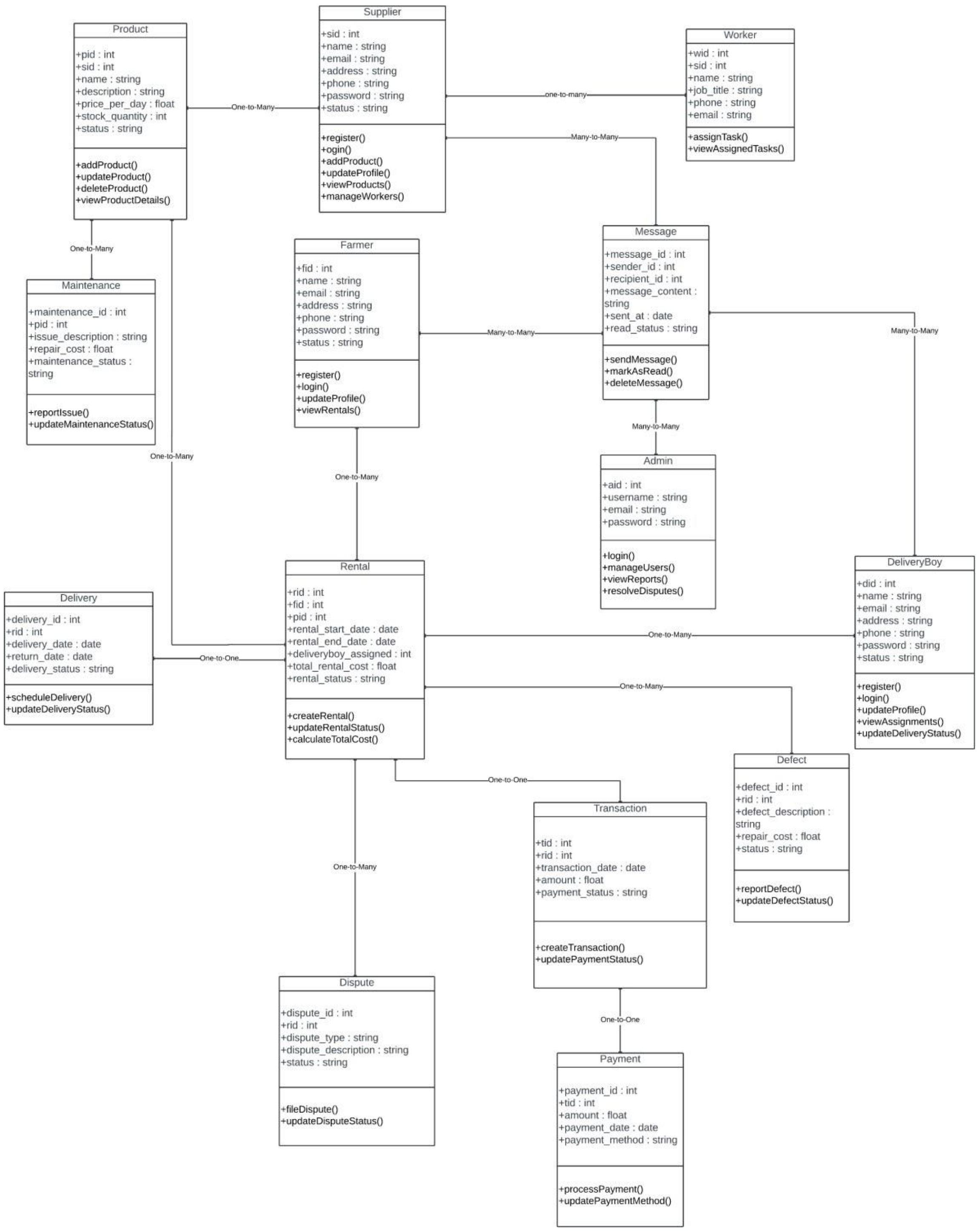

Login
Validate
No
Show Login Error
Yes
Supplier Dashboard Page
Add Product: The supplier adds new products by providing details such as name, description, price per day, stock quantity, and status.
The supplier manages the inventory by updating stock quantities and tracking inventory status.
Handle Maintenance: The supplier manages the maintenance process, including tracking the status and repair cost.
The supplier receives notifications about order updates, defects reported, and maintenance alerts.
Update Product: The supplier updates existing product details.
The supplier views the status of rental orders for their products.
Report Defect: The supplier reports any defects in the products.


Login
Validate
No
Show Login Error
Yes
Deliveryboy Dashboard Home Page
View Assigned Deliveries
Update Delivery Status
Pending to Delivered
Delivered to Returned
Report Defects
Receive Notifications

4.2.5 Class Diagram

A Class Diagram shows the static structure of the system by representing the system's classes, their attributes, methods, and the relationships between the classes.

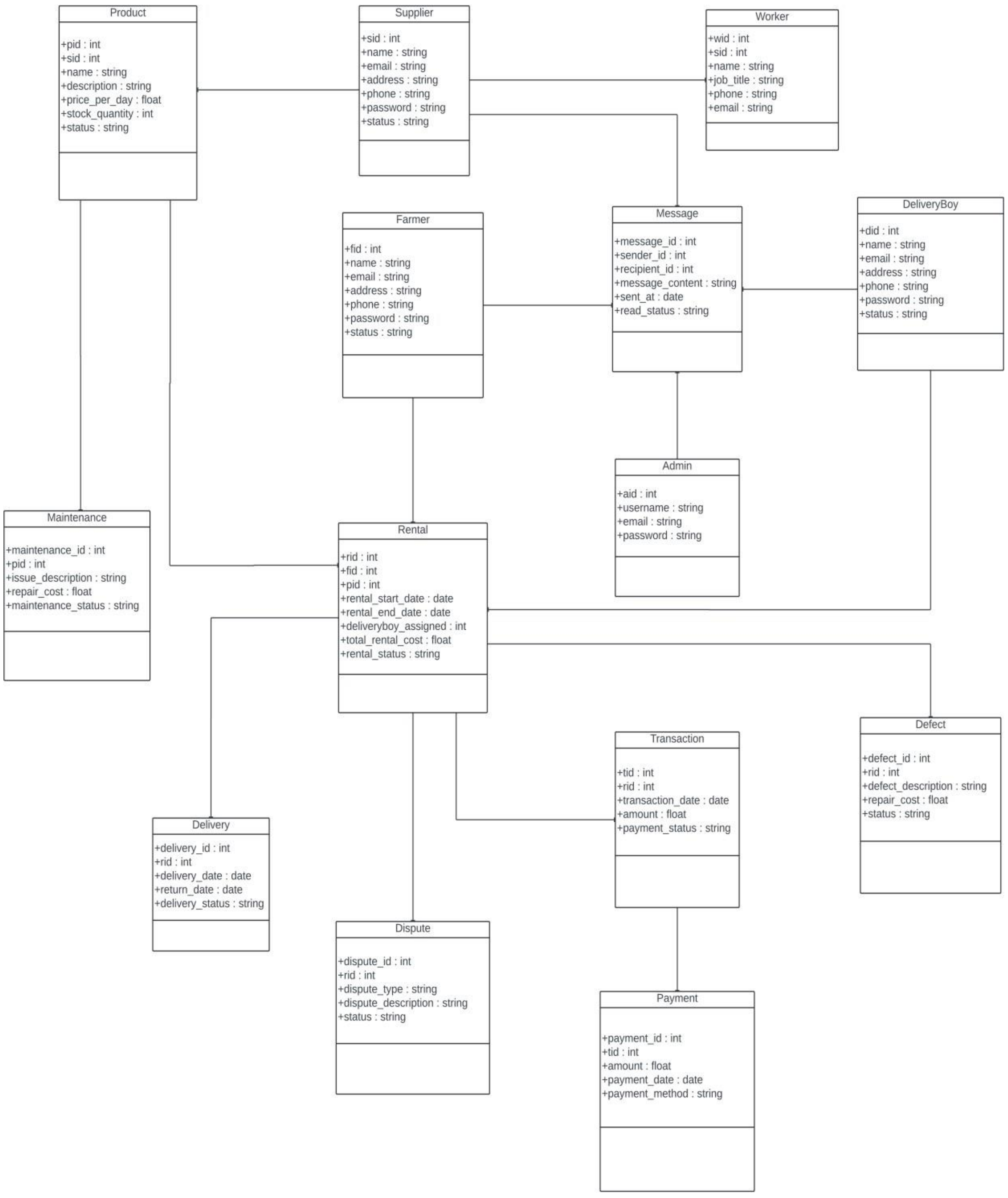
- Example: Classes in AgroRent Hub.
    - Farmer, Supplier, DeliveryBoy, and Admin inherit from the User class.
    - Product class is associated with the Supplier and CartItem classes.
    - Order class is linked to DeliveryBoy and CartItem.



4.2.6 Object Diagram

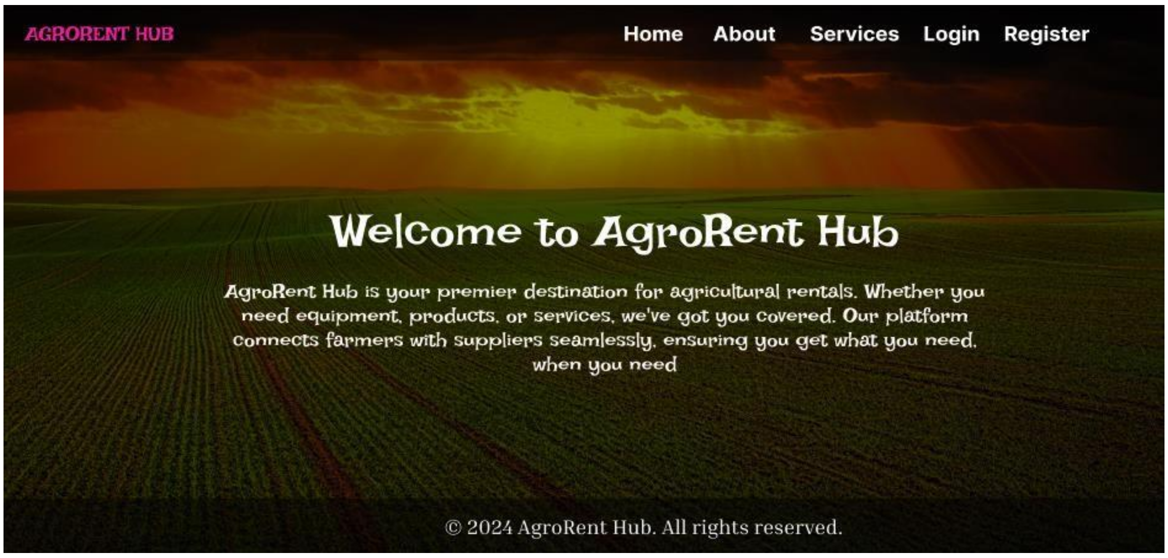
An Object Diagram is a snapshot of the system at a particular moment. It shows instances of classes and the relationships between them.

- Example: A specific order showing the relationship between the Farmer (who placed the order), the Product(s) rented, the assigned Delivery Boy, and the order status.



4.3 USER INTERFACE DESIGN USING FIGMA

Home Page

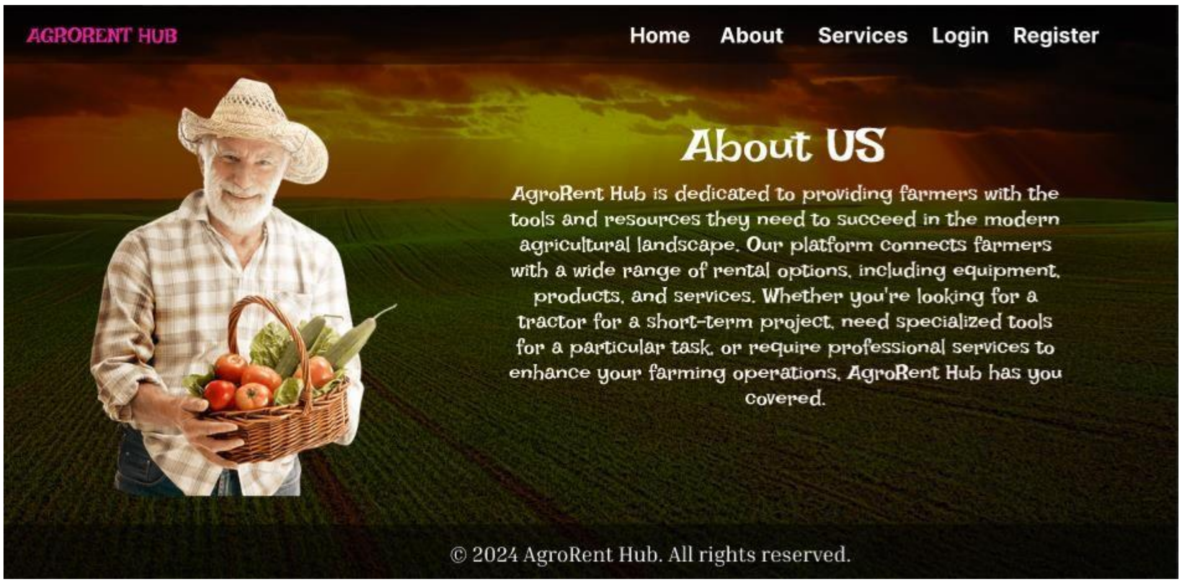


Welcome to AgroRent Hub

AgroRent Hub is your premier destination for agricultural rentals. Whether you need equipment, products, or services, we've got you covered. Our platform connects farmers with suppliers seamlessly, ensuring you get what you need, when you need

© 2024 AgroRent Hub. All rights reserved.

About Us Page

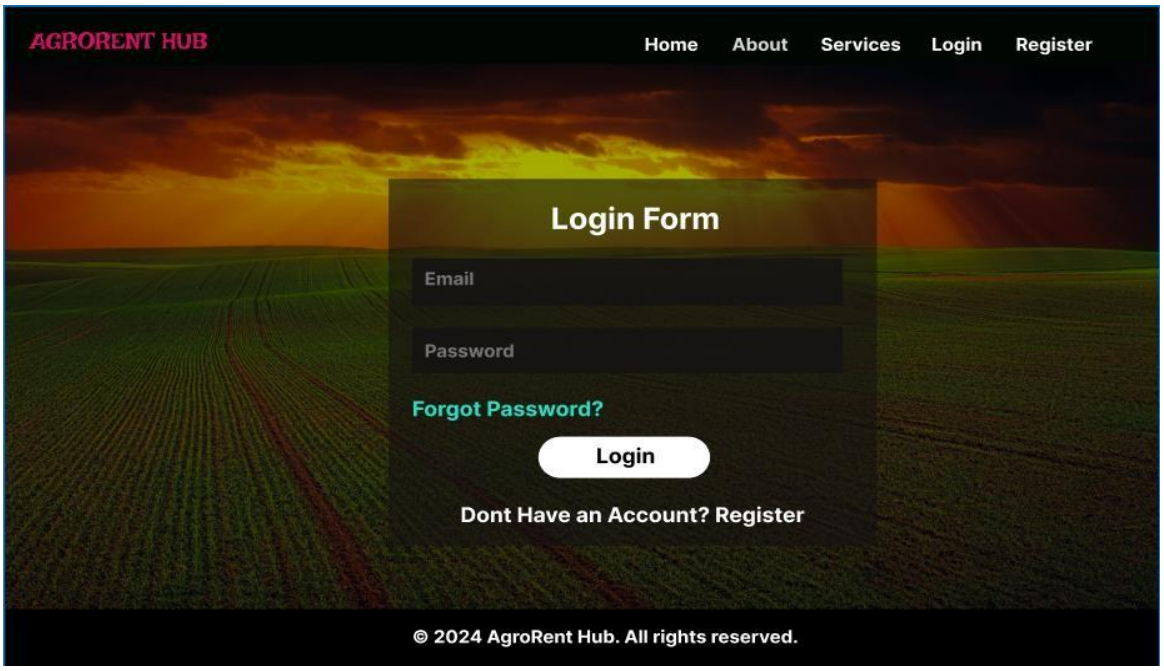


About US

AgroRent Hub is dedicated to providing farmers with the tools and resources they need to succeed in the modern agricultural landscape. Our platform connects farmers with a wide range of rental options, including equipment, products, and services. Whether you're looking for a tractor for a short-term project, need specialized tools for a particular task, or require professional services to enhance your farming operations, AgroRent Hub has you covered.

© 2024 AgroRent Hub. All rights reserved.

Login Form



Login Form

Email

Password

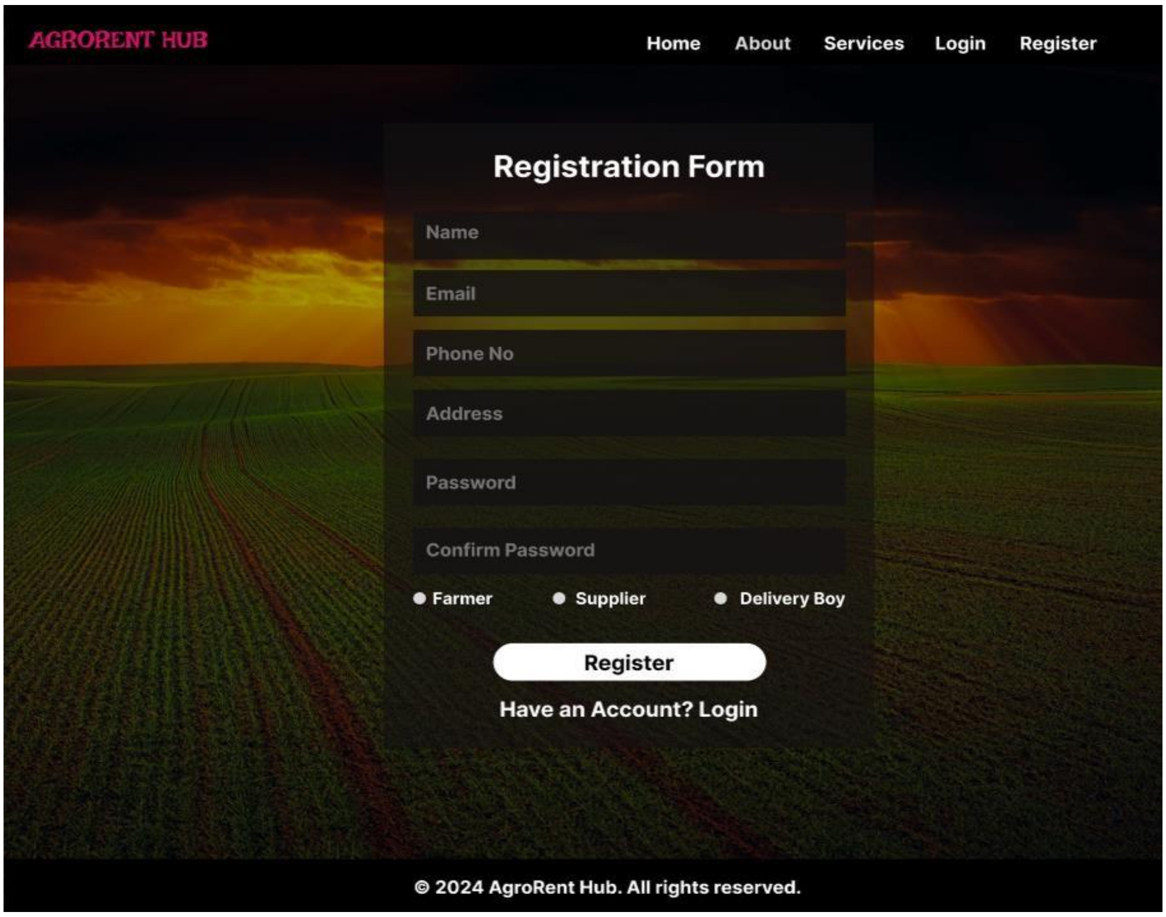
Forgot Password?

Login

Dont Have an Account? Register

© 2024 AgroRent Hub. All rights reserved.

Registration Form



Registration Form

Name

Email

Phone No

Address

Password

Confirm Password

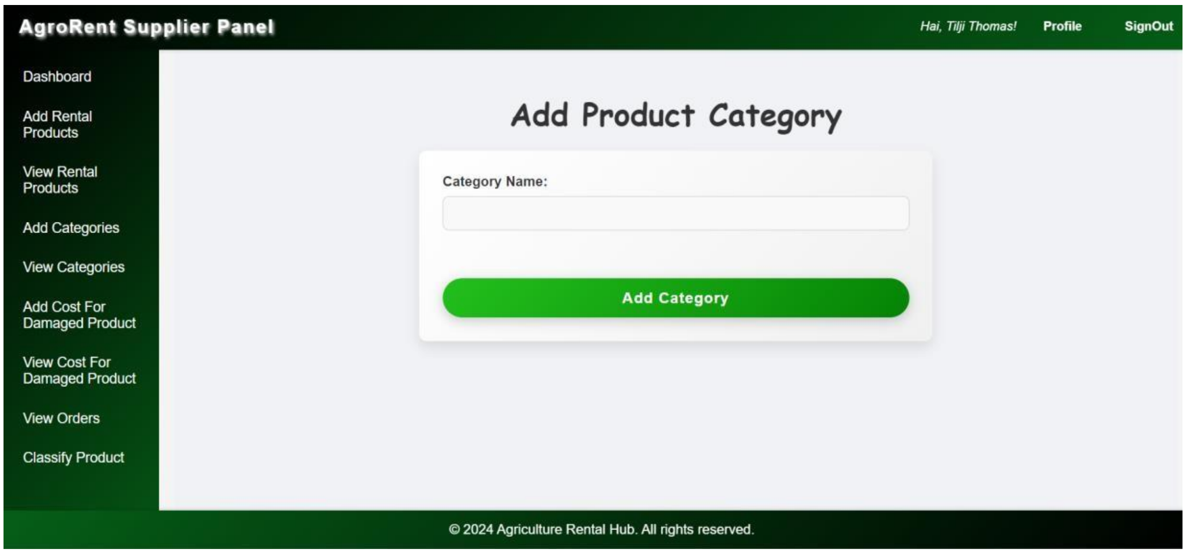
● Farmer ● Supplier ● Delivery Boy

Register

Have an Account? Login

© 2024 AgroRent Hub. All rights reserved.

Add Category Form


AgroRent Supplier Panel
Hai, Tilji Thomas!
Profile
SignOut
Dashboard
Add Rental Products
View Rental Products
Add Categories
View Categories
Add Cost For Damaged Product
View Cost For Damaged Product
View Orders
Classify Product
Add Product Category
Category Name:
Add Category
© 2024 Agriculture Rental Hub. All rights reserved.

4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

Relational Database Management System (RDBMS) is a type of database management system that stores data in tables (relations) and enables users to query and manipulate the data using Structured Query Language (SQL). In AgroRent Hub, the database is designed using an RDBMS to manage structured data, such as user accounts, rental orders, product inventories, and transactions.

- Database used: MySQL (or SQLite for local development).
- Entities: Tables for Farmers, Suppliers, Delivery Boys, Admins, Products, Orders, Workers, Payments, etc.
- Relationships:
  - One-to-many between Suppliers and Products.
  - One-to-many between Farmers and Orders.
  - One-to-one between Orders and Delivery Boys.

4.4.2 Normalization

Normalization is the process of organizing the data in the database to reduce redundancy and ensure data integrity. It involves dividing larger tables into smaller, well-structured tables to avoid anomalies in insertion, deletion, and updating operations.

- First Normal Form (1NF): Ensures that all attributes in a table contain only atomic values (indivisible).
- Second Normal Form (2NF): Ensures that all non-key attributes are fully dependent on the primary key.
- Third Normal Form (3NF): Removes transitive dependencies, ensuring that non-key attributes are not dependent on other non-key attributes.

4.4.3 Sanitization

Sanitization refers to the process of cleaning user inputs to prevent malicious data from being stored in the database, such as SQL injections and cross-site scripting (XSS) attacks. In AgroRent Hub, user inputs are sanitized at multiple stages, including during form submission, before being stored in the database.

- SQL Injection Prevention: Using parameterized queries and Django ORM functions to prevent raw SQL execution.
- Data Validation: Ensuring that user inputs like product names, descriptions, and

rental details conform to expected formats (e.g., no special characters in certain fields).

By sanitizing inputs, the integrity of the database is maintained, and security risks are minimized.

4.4.4 Indexing

Indexing improves the performance of the database by allowing faster retrieval of records. In AgroRent Hub, frequently accessed data (such as products, orders, and user information) is indexed to enhance search efficiency.

- Primary Index: Automatically created on primary key fields such as user_id in the User table or order_id in the Order table.
- Secondary Indexes: Additional indexes can be created on columns frequently used in queries, such as product_name in the Product table or order_status in the Order table.
- Benefits:
  - Improved query performance.
  - Faster access to frequently searched records (e.g., searching products by category or checking order statuses).

4.5 TABLE DESIGN

1. Tbl_Farmer

No:	**Fieldname**	**Datatype (Size)**	**Key Constraints**	**Description of the Field**
1	farmer_id	INT	Primary Key	Unique ID for each farmer
2	name	VARCHAR(100)	Not Null	Farmer's name
3	email	VARCHAR(100)	Unique, Not Null	Farmer's email
4	address	TEXT	Not Null	Farmer's address
5	phone	VARCHAR(15)	Not Null	Farmer's phone number
6	password	VARCHAR(255)	Not Null	Password for authentication
7	status	VARCHAR(10)	Default "active"	Farmer's account status
8	last_login	DATETIME	NULL	Last login time

9	reset_password _token	VARCHAR(100)	NULL	Token for resetting password
10	token_expiry	DATETIME	NULL	Token expiry time
11	reasontoblock	TEXT	NULL	Reason for blocking the account

Primary Key: farmer_id

2. Tbl_Supplier

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	supplier_id	INT	Primary Key	Unique ID for each supplier
2	name	VARCHAR(100)	Not Null	Supplier's name
3	email	VARCHAR(100)	Unique, Not Null	Supplier's email
4	address	TEXT	Not Null	Supplier's address
5	phone	VARCHAR(15)	Not Null	Supplier's phone number
6	password	VARCHAR(255)	Not Null	Password for authentication
7	status	VARCHAR(10)	Default "active"	Supplier's account status
8	last_login	DATETIME	NULL	Last login time
9	reset_password_token	VARCHAR(100)	NULL	Token for resetting password
10	token_expiry	DATETIME	NULL	Token expiry time
11	reasontoblock	TEXT	NULL	Reason for blocking the account

Primary Key: supplier_id

3. Tbl_DeliveryBoy

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field

1	delivery_boy_id	INT	Primary Key	Unique ID for each delivery boy
2	name	VARCHAR(100)	Not Null	Delivery boy's name
3	email	VARCHAR(100)	Unique, Not Null	Delivery boy's email
4	address	TEXT	Not Null	Delivery boy's address
5	phone	VARCHAR(15)	Not Null	Delivery boy's phone number
6	password	VARCHAR(255)	Not Null	Password for authentication
7	status	VARCHAR(10)	Default "active"	Delivery boy's account status
8	last_login	DATETIME	NULL	Last login time
9	reset_password_token	VARCHAR(100)	NULL	Token for resetting password
10	token_expiry	DATETIME	NULL	Token expiry time
11	reasontoblock	TEXT	NULL	Reason for blocking the account

Primary Key: delivery_boy_id

4. Tbl_ProductCategory

No:	**Fieldname**	**Datatype (Size)**	**Key Constraints**	**Description of the Field**
1	category_id	INT	Primary Key	Unique ID for each product category
2	name	VARCHAR(255)	Unique, Not Null	Name of the product category
3	is_disabled	BOOLEAN	Default False	Status of the category

Primary Key: category_id

5. Tbl_Product

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	product_id	INT	Primary Key	Unique ID for each product
2	product_name	VARCHAR(255)	Not Null	Name of the product
3	product_image	VARCHAR(255)	NULL	Image of the product
4	product_price_per_day	DECIMAL(10,2)	Not Null	Price per day for renting the product
5	stock_quantity	INT	Not Null	Available stock of the product
6	product_description	TEXT	Not Null	Description of the product
7	status	VARCHAR(20)	Default "available"	Status of the product
8	is_disabled	BOOLEAN	Default False	Whether the product is disabled
9	supplier_id	INT	Foreign Key	Supplier of the product
10	category_id	INT	Foreign Key	Category of the product

Primary Key: product_id
Foreign Key: supplier_id references Tbl_Supplier(supplier_id)
Foreign Key: category_id references Tbl_ProductCategory(category_id)

6. Tbl_CartItem

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	cart_item_id	INT	Primary Key	Unique ID for each cart item
2	farmer_id	INT	Foreign Key	Farmer who added the item
3	product_id	INT	Foreign Key	Product added to the cart

4	quantity	INT	Default 1	Quantity of the product
5	start_date	DATE	Not Null	Rental start date
6	end_date	DATE	Not Null	Rental end date
7	price	DECIMAL(10,2)	Not Null	Price for the selected period

Primary Key: cart_item_id
Foreign Key: farmer_id references Tbl_Farmer(farmer_id)
Foreign Key: product_id references Tbl_Product(product_id)

7. Tbl_Order

No:	**Fieldname**	**Datatype (Size)**	**Key Constraints**	**Description of the Field**
1	order_id	INT	Primary Key	Unique ID for each order
2	farmer_id	INT	Foreign Key	Farmer who placed the order
3	product_id	INT	Foreign Key	Product in the order
4	delivery_boy_id	INT	Foreign Key, NULL	Delivery boy assigned to the order
5	quantity	INT	Not Null	Quantity of the product ordered
6	start_date	DATE	Not Null	Rental start date
7	end_date	DATE	Not Null	Rental end date
8	price	DECIMAL(10,2)	Not Null	Total price for the rental period
9	ordered_at	DATETIME	Auto-added	Date and time when the order was placed
10	payment_status	VARCHAR(20)	Default "pending"	Payment status (pending/completed)
11	shipping_status	VARCHAR(20)	Default "not shipped"	Shipping status
12	delivery_status	VARCHAR(20)	Default "not delivered"	Delivery status

Primary Key: order_id
Foreign Key: farmer_id references Tbl_Farmer(farmer_id)
Foreign Key: product_id references Tbl_Product(product_id)
Foreign Key: delivery_boy_id references Tbl_DeliveryBoy(delivery_boy_id)

8. Tbl_ReturnRequest

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	return_id	INT	Primary Key	Unique ID for each return request
2	order_id	INT	Foreign Key	Order associated with the return
3	farmer_id	INT	Foreign Key	Farmer requesting the return
4	request_date	DATETIME	Auto-added	Date and time of the return request
5	return_status	VARCHAR(20)	Default "requested"	Status of the return (e.g., requested, picked up, returned)
6	reason_for_return	TEXT	Not Null	Reason for returning the product

Primary Key: return_id
Foreign Key: order_id references Tbl_Order(order_id)
Foreign Key: farmer_id references Tbl_Farmer(farmer_id)

9. ContactUs

Field Name	Data Type	Key Constraints	Description
id	INT	Primary Key	Unique identifier for each contact
name	VARCHAR(100)	Not Null	Name of the person contacting
email	EmailField	Not Null	Email of the person contacting
message	TextField	Not Null	Message from the user
created_at	DateTime	Auto-added	Date the message was submitted

10. DamageLevel

Field Name	Data Type	Key Constraints	Description
damage_level_id	INT	Primary Key	Unique ID for each level
damage_level	VARCHAR(20)	Not Null	Level of damage (e.g., minor, moderate, severe)

11. DamageType

Field Name	Data Type	Key Constraints	Description
damage_type_id	INT	Primary Key	Unique ID for each type
damage_type	VARCHAR(20)	Not Null	Type of damage (e.g., physical, operational)

12. ProductDamage

Field Name	Data Type	Key Constraints	Description
id	INT	Primary Key	Unique identifier for each damage report
product_id	ForeignKey(Product)	Foreign Key	The product that has been damaged
damage_level_id	ForeignKey(DamageLevel)	Foreign Key	Reference to damage level (e.g., minor, severe)
damage_type_id	ForeignKey(DamageType)	Foreign Key	Reference to damage type (e.g., physical, operational)
estimated_repair_cost	DECIMAL(10,2)	Not Null	Estimated repair cost for the product
created_at	DateTime	Auto-added	Date when the damage report was created

Primary Key: id

Foreign Key: product_id references Product(product_id)

Foreign Key: damage_level_id references DamageLevel(damage_level_id)

Foreign Key: damage_type_id references DamageType(damage_type_id)

13. ReturnRequest

Field Name	Data Type	Key Constraints	Description
return_id	INT	Primary Key	Unique identifier for

			each return request
order_id	ForeignKey(Order)	Foreign Key	Order associated with the return
product_id	ForeignKey(Product)	Foreign Key	Product associated with the return request
quantity	INT	Not Null	Quantity of the product being returned
damage_level_id	ForeignKey(DamageLevel)	Foreign Key	Reference to damage level (e.g., minor, severe)
damage_type_id	ForeignKey(DamageType)	Foreign Key	Reference to damage type (e.g., physical, operational)
estimated_repair_cost	DECIMAL(10,2)	Not Null	Estimated repair cost for the product
total_repair_cost	DECIMAL(10,2)	Default 0.0	Total repair cost after evaluation
razorpay_payment_id	VARCHAR(255)	NULL	Razorpay Payment ID for return request (if any)
created_at	DateTime	Auto-added	Date when the return request was created

Primary Key: return_id

Foreign Key: order_id references Order(order_id)

Foreign Key: product_id references Product(product_id)

Foreign Key: damage_level_id references DamageLevel(damage_level_id)

Foreign Key: damage_type_id references DamageType(damage_type_id)

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

System testing is a crucial phase in the software development lifecycle that focuses on validating the entire system's functionality, performance, and reliability. It aims to ensure that the application meets both functional and non-functional requirements as outlined in the specifications. By executing tests on the complete integrated system, the goal is to identify any discrepancies between actual and expected outcomes. This process helps to uncover defects that might not have been evident during earlier testing stages, ultimately ensuring that the software is ready for deployment and meets user expectations.

5.2 TEST PLAN

A test plan is a strategic document that outlines the scope, approach, resources, and schedule for testing activities throughout the software development process. It serves as a roadmap for testing, detailing the objectives, methodologies, and tools that will be employed to validate the software. The test plan also defines the roles and responsibilities of the testing team, along with the criteria for evaluating success. By providing a clear framework for the testing process, the test plan helps ensure comprehensive coverage and facilitates effective communication among team members and stakeholders.

5.2.1 Unit Testing

Unit testing is a fundamental testing methodology that focuses on evaluating individual components or modules of a software application in isolation. The primary objective of unit testing is to verify that each unit of code performs as intended, ensuring that functions or methods yield the expected results. This testing phase is typically carried out by developers during the coding phase to catch and address issues early in the development process. Unit tests are generally automated, utilizing frameworks such as JUnit for Java, NUnit for .NET, or PyTest for Python, allowing for efficient and consistent testing of individual code units.

5.2.2 Integration Testing

Integration testing is the next step after unit testing, focusing on evaluating the interactions and interfaces between different modules or components of the system. The primary goal is to identify issues that may arise when integrating various parts of the application, ensuring they work together as intended. This testing phase can employ different approaches, such as top-down, bottom-up, or a hybrid method, depending on the architecture of the software. Developers and testers usually conduct integration tests to verify that combined components function correctly and that data flows seamlessly across the integrated modules.

5.2.3 Validation Testing or System Testing

Validation testing, also known as system testing, is a comprehensive testing phase that evaluates the entire system's functionality and performance. It aims to ensure that the software meets all specified requirements, as well as any applicable regulatory or compliance standards. During this phase, QA teams execute a series of end-to-end tests on the fully integrated system to validate the overall behavior of the application. Validation testing is essential for identifying defects and discrepancies before the software is released, providing confidence that the system will perform reliably in real-world conditions.

5.2.4 Output Testing or User Acceptance Testing

User Acceptance Testing (UAT) is the final testing phase that focuses on validating the software from the end user’s perspective. It ensures that the application meets the business needs and requirements as defined by the users. During UAT, real users test the system in a production-like environment to verify that it behaves as expected and is user-friendly. This phase is critical for ensuring customer satisfaction, as it provides an opportunity to gather feedback and make necessary adjustments before the software goes live. UAT is typically conducted by business stakeholders or end users who are familiar with the application’s intended functionality.

5.2.5 Automation Testing

Automation testing leverages tools and scripts to execute tests automatically, streamlining the testing process and enhancing efficiency. This methodology is particularly beneficial for repetitive tasks, such as regression testing, where the same tests need to be run multiple times across different iterations of the software. Automation testing helps increase test coverage and accuracy while reducing the time and effort required for manual testing. It is commonly used by testers with expertise in automation tools like Selenium, QTP, and TestComplete, enabling faster feedback cycles and quicker identification of defects.

5.2.6 Selenium Testing

Selenium testing is a specialized form of automation testing specifically designed for web applications. It utilizes Selenium, an open-source tool that allows testers to create scripts that automate browser interactions. The primary goal of Selenium testing is to validate the behavior of web applications across various browsers and platforms, ensuring that they function correctly in different environments. Testers can write test scripts in multiple programming languages, such as Java, Python, or C#, and use Selenium WebDriver to simulate user actions like clicking buttons, filling out forms, and navigating through web pages. By automating these interactions, Selenium testing enhances the efficiency and reliability of the testing process, making it easier to maintain high-quality web applications.

Example:

Test Case 1: Login Test Case

Code

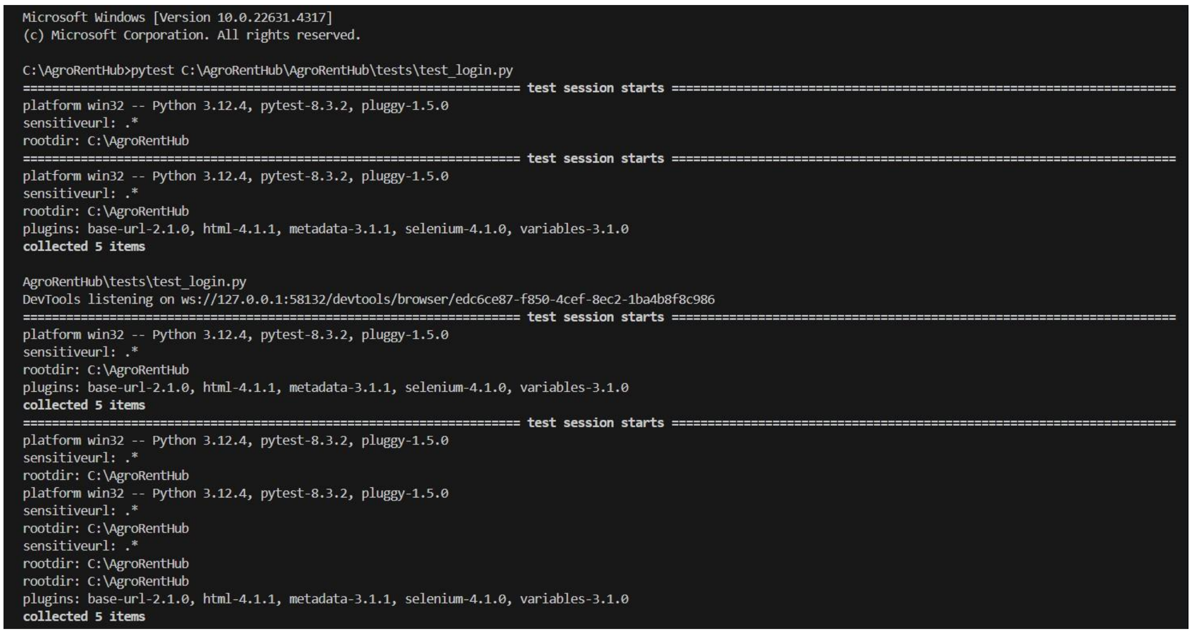
```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.webdriver import WebDriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException  # Ensure this
import is correct
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

@pytest.fixture
def browser():
    chrome_options = Options()
    # chrome_options.add_argument("--headless")  # Ensure GUI is on
    chrome_options.add_argument("--no-sandbox")
    chrome_options.add_argument("--disable-dev-shm-usage")

    # Path to Brave Browser executable
    brave_path = r"C:\Users\wilgi\AppData\Local\BraveSoftware\Brave-
Browser\Application\brave.exe"  # Use raw string for Windows paths
    chrome_options.binary_location = brave_path

    # Use WebDriver Manager to get the correct version of chromedriver
    service = Service(ChromeDriverManager().install())

    driver = webdriver.Chrome(service=service, options=chrome_options)
    driver.maximize_window()  # Maximize the window to full screen
    yield driver
    driver.quit()

def type_slowly(element, text, delay=0.1):
    """
    Type text into an input element with a delay between each keystroke.
    """
    for character in text:
        element.send_keys(character)
        time.sleep(delay)

def test_login_correct_credentials(browser: WebDriver):
    # Navigate to login page
    browser.get('http://localhost:8000/login')  # Replace with your local
server URL

    # Find email and password fields
    email_field = browser.find_element(By.NAME, 'email')
    password_field = browser.find_element(By.NAME, 'password')

    # Enter login credentials slowly
    type_slowly(email_field, 'wilgimolthomas@gmail.com')
    type_slowly(password_field, 'Wilgi@123')

    # Submit the form
    password_field.send_keys(Keys.RETURN)

    # Verify successful login by checking for the presence of an element on
the customer_index page
    assert 'customer_index' in browser.current_url  # Ensure you're
redirected correctly

    # Slowly scroll down the page
    scroll_pause_time = 0.5  # Pause time between scrolls (in seconds)
    screen_height = browser.execute_script("return window.innerHeight;")
    scroll_height = browser.execute_script("return
document.body.scrollHeight;")

    for i in range(0, scroll_height, screen_height):
        browser.execute_script(f"window.scrollTo(0, {i});")
        time.sleep(scroll_pause_time)

    # Optionally, wait for a few seconds at the end to observe the fully
scrolled page
    time.sleep(2)

    # Verify that the entire page is scrolled by checking for an element at
the bottom
    footer = browser.find_element(By.TAG_NAME, 'footer')
    assert footer.is_displayed()

def test_login_admin(browser: WebDriver):
    # Navigate to login page
    browser.get('http://localhost:8000/login')  # Replace with your local
server URL
```

```
    # Find email and password fields
    email_field = browser.find_element(By.NAME, 'email')
    password_field = browser.find_element(By.NAME, 'password')

    # Enter admin login credentials slowly
    type_slowly(email_field, 'admin@gmail.com')
    type_slowly(password_field, 'admin@123')

    # Submit the form
    password_field.send_keys(Keys.RETURN)

    # Verify successful login by checking for the presence of an element on
the admin_index page
    assert 'admin_index' in browser.current_url  # Ensure you're redirected
correctly

    # Optionally, wait for a few seconds to observe the fully loaded page
    time.sleep(2)

def test_login_supplier(browser: WebDriver):
    # Navigate to login page
    browser.get('http://localhost:8000/login')  # Replace with your local
server URL

    # Find email and password fields
    email_field = browser.find_element(By.NAME, 'email')
    password_field = browser.find_element(By.NAME, 'password')

    # Enter supplier login credentials slowly
    type_slowly(email_field, 'tiljithomas@gmail.com')
    type_slowly(password_field, 'Tilji@1234')

    # Submit the form
    password_field.send_keys(Keys.RETURN)

    # Verify successful login by checking for the presence of an element on
the supplier_index page
    assert 'supplier_index' in browser.current_url  # Ensure you're
redirected correctly

    # Optionally, wait for a few seconds to observe the fully loaded page
    time.sleep(2)

def test_login_deliveryboy(browser: WebDriver):
    # Navigate to login page
    browser.get('http://localhost:8000/login')  # Replace with your local
server URL
```

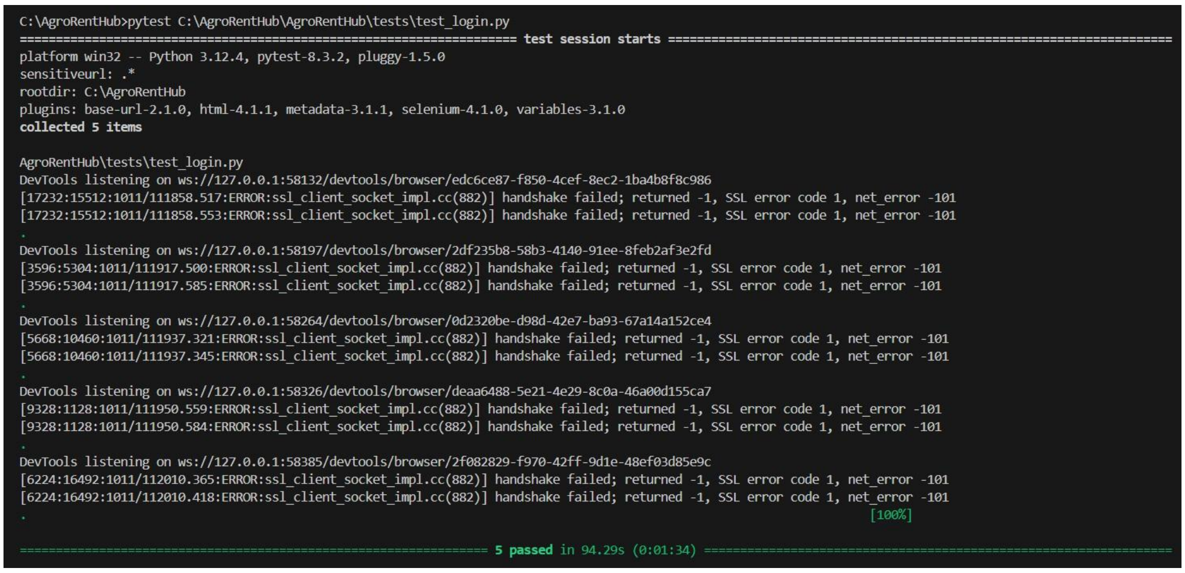
```
    # Find email and password fields
    email_field = browser.find_element(By.NAME, 'email')
    password_field = browser.find_element(By.NAME, 'password')

    # Enter deliveryboy login credentials slowly
    type_slowly(email_field, 'jiltithomas@gmail.com')
    type_slowly(password_field, 'Jilti@123')

    # Submit the form
    password_field.send_keys(Keys.RETURN)

    # Verify successful login by checking for the presence of an element on
the deliveryboy_index page
    assert 'deliveryboy_index' in browser.current_url  # Ensure you're
redirected correctly

    # Optionally, wait for a few seconds to observe the fully loaded page
    time.sleep(2)


def test_login_incorrect_credentials(browser: WebDriver):
    # Navigate to login page
    browser.get('http://localhost:8000/login')  # Replace with your local
server URL

    # Find email and password fields
    email_field = browser.find_element(By.NAME, 'email')
    password_field = browser.find_element(By.NAME, 'password')

    # Enter incorrect login credentials slowly
    type_slowly(email_field, 'wrongemail@example.com')
    type_slowly(password_field, 'WrongPassword')

    # Submit the form
    password_field.send_keys(Keys.RETURN)

    # Verify that the URL is still the login page
    assert 'login' in browser.current_url  # Ensure you're still on the
login page

    # Look for the error message element without explicit waiting
    error_message = WebDriverWait(browser, 10).until(
    EC.visibility_of_element_located((By.CLASS_NAME, 'error-message'))
)

    # Assertions
    assert error_message.is_displayed()
    assert 'Invalid email or password' in error_message.text
```

```python
    # Optionally, wait for a few seconds to observe the error message
    time.sleep(2)
```

ScreenShot



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\AgroRentHub>pytest C:\AgroRentHub\AgroRentHub\tests\test_login.py
================================================================= test session starts ==================================================================
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
================================================================= test session starts ==================================================================
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
plugins: base-url-2.1.0, html-4.1.1, metadata-3.1.1, selenium-4.1.0, variables-3.1.0
collected 5 items

AgroRentHub\tests\test_login.py
DevTools listening on ws://127.0.0.1:58132/devtools/browser/edc6ce87-f850-4cef-8ec2-1ba4b8f8c986
================================================================= test session starts ==================================================================
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
plugins: base-url-2.1.0, html-4.1.1, metadata-3.1.1, selenium-4.1.0, variables-3.1.0
collected 5 items
================================================================= test session starts ==================================================================
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
sensitiveurl: .*
rootdir: C:\AgroRentHub
rootdir: C:\AgroRentHub
plugins: base-url-2.1.0, html-4.1.1, metadata-3.1.1, selenium-4.1.0, variables-3.1.0
collected 5 items
```



```
C:\AgroRentHub>pytest C:\AgroRentHub\AgroRentHub\tests\test_login.py
================================================================= test session starts ==================================================================
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
plugins: base-url-2.1.0, html-4.1.1, metadata-3.1.1, selenium-4.1.0, variables-3.1.0
collected 5 items

AgroRentHub\tests\test_login.py
DevTools listening on ws://127.0.0.1:58132/devtools/browser/edc6ce87-f850-4cef-8ec2-1ba4b8f8c986
[17232:15512:1011/111858.517:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[17232:15512:1011/111858.553:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
.
DevTools listening on ws://127.0.0.1:58197/devtools/browser/2df235b8-58b3-4140-91ee-8feb2af3e2fd
[3596:5304:1011/111917.500:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[3596:5304:1011/111917.585:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
.
DevTools listening on ws://127.0.0.1:58264/devtools/browser/0d2320be-d98d-42e7-ba93-67a14a152ce4
[5668:10460:1011/111937.321:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[5668:10460:1011/111937.345:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
.
DevTools listening on ws://127.0.0.1:58326/devtools/browser/deaa6488-5e21-4e29-8c0a-46a00d155ca7
[9328:1128:1011/111950.559:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[9328:1128:1011/111950.584:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
.
DevTools listening on ws://127.0.0.1:58385/devtools/browser/2f082829-f970-42ff-9d1e-48ef03d85e9c
[6224:16492:1011/112010.365:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[6224:16492:1011/112010.418:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
.                                                                                                                       [100%]

============================================================== 5 passed in 94.29s (0:01:34) ==============================================================
```

Test Report

Test Case 1

Project Name: AgroRent Hub	
Login Test Case	
Test Case ID: Test_1	**Test Designed By: Wilgimol Thomas**
Test Priority(Low/Medium/High):High	**Test Designed Date: 20/09/2024**
Module Name: Login	**Test Executed By : G. S Ajith**

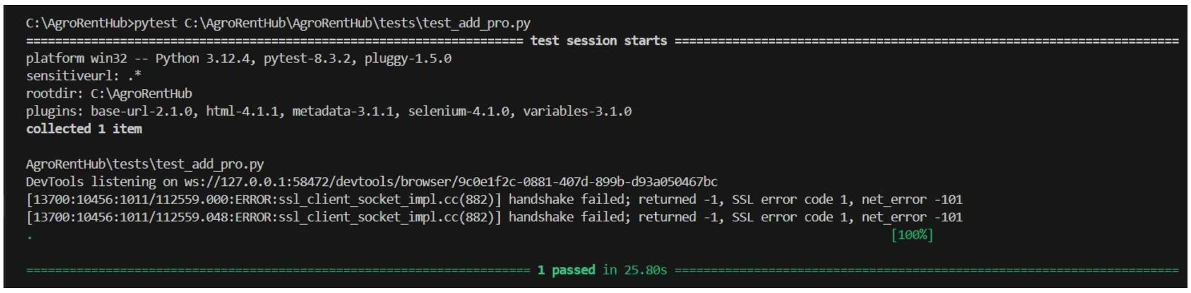
Test Title : Successful Login Test			**Test Execution Date: 11/10/2024**		
Description: Validate that users can log in with valid credentials.					
Pre-Condition : User has a valid username and password.					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigate to the login page	N/A	Login page is displayed.	Login page is displayed successfully.	Pass
2	Enter valid username	Wilgimolthomas@gmail.com	Username is accepted.	Username is accepted without errors.	Pass
3	Enter valid password	Wilgi@123	Password is accepted.	Password is accepted without errors.	
4	Click on the 'Login' button	N/A	User is redirected to the dashboard/homepage.	User is redirected to the dashboard successfully.	
5	Verify the welcome message	N/A	Welcome message displays user's name.	Welcome message "Welcome, User!" is displayed.	Pass
6	Check if the user is logged in	N/A	User session is active.	User session is active with user information.	
7	Log out from the application	N/A	User is logged out and redirected to the login page.	User is logged out and redirected successfully.	
Post-Condition: User should be able to log in and log out successfully.					

Test Case 2: Add Product Test Case

Code

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

@pytest.fixture
def browser():
    chrome_options = Options()
    chrome_options.add_argument("--no-sandbox")
    chrome_options.add_argument("--disable-dev-shm-usage")

    # Path to Brave Browser executable if you're using Brave
    brave_path = r"C:\Users\wilgi\AppData\Local\BraveSoftware\Brave-
Browser\Application\brave.exe"
    chrome_options.binary_location = brave_path

    # Use WebDriver Manager to get the correct version of chromedriver
    service = Service(ChromeDriverManager().install())

    driver = webdriver.Chrome(service=service, options=chrome_options)
    driver.maximize_window()  # Maximize the browser window
    yield driver
    driver.quit()

def type_slowly(element, text, delay=0.1):
    """Type text into an input element with a delay between each
keystroke."""
    for character in text:
        element.send_keys(character)
        time.sleep(delay)

def test_add_rental_product(browser):
    # Open the login page
    browser.get('http://127.0.0.1:8000/login')  # Replace with your server
URL

    # Find the login fields and enter credentials
    email_field = browser.find_element(By.NAME, 'email')
    password_field = browser.find_element(By.NAME, 'password')

    # Enter login credentials
    type_slowly(email_field, 'tiljithomas@gmail.com')
    type_slowly(password_field, 'Tilji@1234')

    # Submit the login form
    password_field.send_keys(Keys.RETURN)

    # Wait for the dashboard to load
    WebDriverWait(browser, 10).until(EC.url_contains('/supplier_index'))
    # Navigate to the "Add Rental Products" page
    browser.get('http://127.0.0.1:8000/supplier_add_pro/')  # Replace with
your server URL

    # Wait for the form to be visible
    WebDriverWait(browser,
10).until(EC.visibility_of_element_located((By.TAG_NAME,  'form')))

    # Fill out the form fields
    product_name = browser.find_element(By.ID, 'product_name')
    product_price_per_day = browser.find_element(By.ID,
'product_price_per_day')
    stock_quantity = browser.find_element(By.ID, 'stock_quantity')
    product_description = browser.find_element(By.ID, 'product_description')
    status = browser.find_element(By.ID, 'status')
    category = browser.find_element(By.ID, 'category')
    product_image = browser.find_element(By.ID, 'product_image')

    # Input product details
    type_slowly(product_name, 'Test Product New')
    type_slowly(product_price_per_day, '100.00')
    type_slowly(stock_quantity, '10')
    type_slowly(product_description, 'This is a test product description.')

    # Select a status
    status.click()
    status.find_element(By.XPATH, "//option[text()='Available']").click()  #
Choose 'Available'

    # Select a category
    category.click()
    category.find_element(By.XPATH, "//option[text()='Harvesting
Equipment']").click()  # Choose the first category

    # Upload a product image (ensure you have a valid image path)
    product_image.send_keys(r"C:\Users\wilgi\Downloads\Balers2.jpeg")  # No
quotes around the path

    # Check if product already exists before submitting
    # Navigate to the product view page to check for existing products
    browser.get('http://127.0.0.1:8000/supplier_view_pro/')  # Replace with
your server URL
    WebDriverWait(browser,
10).until(EC.visibility_of_element_located((By.TAG_NAME, 'table')))  #
Adjust if needed

    # Look for the product in the existing product table
    existing_products = browser.find_elements(By.XPATH, "//table//tr")  #
Adjust the XPath to match your table rows
    product_exists = False

    for row in existing_products:
        if "Test Product New" in row.text:  # Adjust the text to match how
the product is displayed
            product_exists = True
            break

    if product_exists:
        print("Product already exists. Form will not be submitted.")
        assert True # Optionally, assert something meaningful
    else:
        # Submit the form
        form = browser.find_element(By.TAG_NAME, 'form')
        form.submit()

        # Wait for the page to load after submission
        WebDriverWait(browser,
10).until(EC.url_contains('/supplier_view_pro'))

        # Debugging print to check the page source
        print("Page Source after submission:")
        print(browser.page_source)  # Print the current page source

        # Optionally, check for success message or image presence
        try:
            success_message = WebDriverWait(browser,
10).until(EC.visibility_of_element_located((By.XPATH,
"//div[contains(text(), 'Product added successfully')]")))
            assert success_message.is_displayed()
        except TimeoutException:  # type: ignore
            print("Success message not found. Check the page source or the
XPath.")
            raise  # Re-raise the exception to see the original error
message

        # Assert the success of the submission by checking if redirected to
the view products page
        assert "View Rental Products" in browser.page_source
```

Screenshot



```
C:\AgroRentHub>pytest C:\AgroRentHub\AgroRentHub\tests\test_add_pro.py
======================================================================= test session starts ========================================================================
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
plugins: base-url-2.1.0, html-4.1.1, metadata-3.1.1, selenium-4.1.0, variables-3.1.0
collected 1 item

AgroRentHub\tests\test_add_pro.py
DevTools listening on ws://127.0.0.1:58472/devtools/browser/9c0e1f2c-0881-407d-899b-d93a050467bc
[13700:10456:1011/112559.000:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[13700:10456:1011/112559.048:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
.                                                                                                                                    [100%]

======================================================================== 1 passed in 25.80s ========================================================================
```

Test Report

Test Case 2

Project Name:AgroRent Hub	
Add Product Test Case	
Test Case ID: Test_2	**Test Designed By: Wilgimol Thomas**
Test Priority(Low/Medium/High):High	**Test Designed Date: 08/10/2024**
Module Name: Add Product	**Test Executed By : G. S Ajith**
Test Title : Add New Product	**Test Execution Date: 11/10/2024**
Description: This test case verifies that a user can successfully add a new product to the system.	
Pre-Condition : User has valid username and password and is logged into the admin dashboard.	

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigate to the Add Product page	URL: http://127.0.0.1:8000/admin/add_product/	The Add Product page should load successfully.	The Add Product page loaded successfully.	Pass
2	Fill in the product name	Product Name: "New Product"	The product name field should accept the input.	Product name accepted as "New Product."	Pass
3	Fill in the product description	Description: "This is a test product."			
4	Enter the product price	Price: 500	The product description field should accept the input.	Product description accepted.	
5	Upload a product image	Image File: product_image .png	The image should upload successfully.	Image uploaded successfully.	Pass

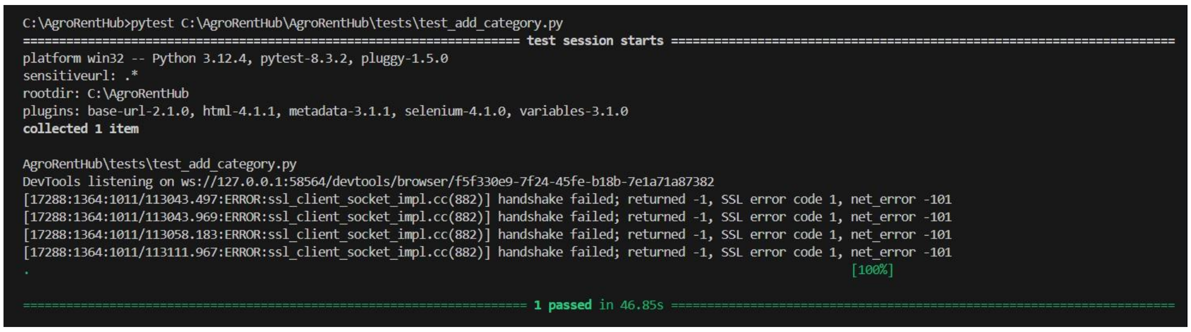
6	Click on the Submit button		The product should be added to the database.	Product "New Product" added to the database.	
7	Check for a success message on the page		A success message should be displayed.	Success message: "Product added successfully."	
Post-Condition: The new product is successfully added to the database and is available in the product list.					

Test Case 3: Add Category Test Case

Code

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

@pytest.fixture
def browser():
    chrome_options = Options()
    chrome_options.add_argument("--no-sandbox")
    chrome_options.add_argument("--disable-dev-shm-usage")

    # Path to Brave Browser executable if you're using Brave
    brave_path = r"C:\Users\wilgi\AppData\Local\BraveSoftware\Brave-
Browser\Application\brave.exe"
    chrome_options.binary_location = brave_path

    # Use WebDriver Manager to get the correct version of chromedriver
    service = Service(ChromeDriverManager().install())

    driver = webdriver.Chrome(service=service, options=chrome_options)
    driver.maximize_window()  # Maximize the browser window
    yield driver
    driver.quit()

def type_slowly(element, text, delay=0.1):
    """
    Type text into an input element with a delay between each keystroke.
    """
    for character in text:
        element.send_keys(character)
        time.sleep(delay)

def test_add_category(browser):
    # Open the login page
    browser.get('http://127.0.0.1:8000/login')  # Replace with your server
URL

    # Find the login fields and enter credentials
    email_field = browser.find_element(By.NAME, 'email')
    password_field = browser.find_element(By.NAME, 'password')

    # Enter login credentials
    type_slowly(email_field, 'tiljithomas@gmail.com')
    type_slowly(password_field, 'Tilji@1234')

    # Submit the login form
    password_field.send_keys(Keys.RETURN)

    # Wait for the dashboard page to load
    try:
        WebDriverWait(browser,
20).until(EC.presence_of_element_located((By.LINK_TEXT, 'Classify
Product')))
    except Exception as e:
        print(f"Error: {e}")
        raise

    # Pause for 5 seconds after login
    time.sleep(5)

    # Navigate to "Add Categories" page
    try:
        browser.find_element(By.LINK_TEXT, "Add Categories").click()
        WebDriverWait(browser,
10).until(EC.presence_of_element_located((By.ID, 'categoryForm')))
    except Exception as e:
        print(f"Error: {e}")
        raise

    # Find the category name input and error message element
    category_name_input = browser.find_element(By.ID, 'name')
    name_error = browser.find_element(By.ID, 'name-error')

    # Test case 1: Invalid category name (less than 3 characters)
    category_name_input.clear()
    category_name_input.send_keys("ab")
    category_name_input.send_keys(Keys.TAB)  # Trigger blur event
    WebDriverWait(browser, 5).until(EC.text_to_be_present_in_element((By.ID, 
'name-error'), "Category name must be between 3 to 50 characters."))
    assert "Category name must be between 3 to 50 characters." in 
name_error.text

    # Pause for 5 seconds after the first validation
    time.sleep(5)

    # Test case 2: Invalid category name (contains numbers or special 
characters)
    category_name_input.clear()
    category_name_input.send_keys("123@#")
    category_name_input.send_keys(Keys.TAB)  # Trigger blur event
    WebDriverWait(browser, 5).until(EC.text_to_be_present_in_element((By.ID, 
'name-error'), "Category name can only contain letters and spaces."))
    assert "Category name can only contain letters and spaces." in 
name_error.text

    # Pause for 5 seconds after the second validation
    time.sleep(5)

    # Test case 3: Valid category name, but category already exists
    category_name_input.clear()
    category_name_input.send_keys("Harvesting Equipment")
    category_name_input.send_keys(Keys.TAB)  # Trigger blur event
    WebDriverWait(browser, 5).until(EC.text_to_be_present_in_element((By.ID, 
'name-error'), "Category name already exists."))
    assert "Category name already exists." in name_error.text

    # Pause for 5 seconds after the third validation
    time.sleep(5)

    # Test case 4: Valid category name, doesn't exist
    new_category = "New Equipment Category"
    category_name_input.clear()
    category_name_input.send_keys(new_category)
    category_name_input.send_keys(Keys.TAB)  # Trigger blur event

    # Wait for the error message to disappear (indicating it's a valid 
input)
    WebDriverWait(browser, 
5).until(EC.invisibility_of_element_located((By.ID, 'name-error')))

    # Pause for 5 seconds after the fourth validation
    time.sleep(5)

    # Submit the form
    submit_button = browser.find_element(By.CSS_SELECTOR,
'button[type="submit"]')
    submit_button.click()

    # Confirm successful submission (you can check for a success message or
redirect)
    WebDriverWait(browser, 10).until(EC.url_contains('supplier_add_cat'))
    assert "supplier_add_cat" in browser.current_url

    # Pause for 5 seconds after successful submission
    time.sleep(5)
```

Screenshot



```
C:\AgroRentHub>pytest C:\AgroRentHub\AgroRentHub\tests\test_add_category.py
====================================================================== test session starts ======================================================================
platform win32 -- Python 3.12.4, pytest-8.3.2, pluggy-1.5.0
sensitiveurl: .*
rootdir: C:\AgroRentHub
plugins: base-url-2.1.0, html-4.1.1, metadata-3.1.1, selenium-4.1.0, variables-3.1.0
collected 1 item

AgroRentHub\tests\test_add_category.py
DevTools listening on ws://127.0.0.1:58564/devtools/browser/f5f330e9-7f24-45fe-b18b-7e1a71a87382
[17288:1364:1011/113043.497:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[17288:1364:1011/113043.969:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[17288:1364:1011/113058.183:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
[17288:1364:1011/113111.967:ERROR:ssl_client_socket_impl.cc(882)] handshake failed; returned -1, SSL error code 1, net_error -101
.                                                                                                                             [100%]

====================================================================== 1 passed in 46.85s ======================================================================
```

Test Report

Test Case 3					
Project Name:AgroRent Hub					
Add Category Test Case					
Test Case ID: Test_3			**Test Designed By: Wilgimol Thomas**		
Test Priority(Low/Medium/High): High			**Test Designed Date: 08/10/2024**		
Module Name: Add Category			**Test Executed By : G. S Ajith**		
Test Title :			**Test Execution Date: 11/10/2024**		
Description: This test case verifies that a user can successfully add a new category to the system.					
Pre-Condition : User has valid username and password and is logged into the admin dashboard.					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**

1	Navigate to the Add Category page	URL: http://127.0.0.1:8000/admin/add_category/	The Add Category page should load successfully.	The Add Category page loaded successfully.	Pass
2	Fill in the category name	Category Name: "New Category"	The category name field should accept the input.	Category name accepted as "New Category."	Pass
3	Click on the Submit button		The category should be added to the database.	Category "New Category" added to the database.	
4	Check for a success message on the page		A success message should be displayed.	Success message: "Category added successfully."	
5	Verify the new category appears in the category list		The new category should be visible in the category list.	New category "New Category" is visible in the category list.	Pass
6	Attempt to add a category with a duplicate name	Category Name: "New Category"	An error message should indicate the category already exists.	Error message displayed: "Category already exists."	
7	Check if the category can be deleted	Category Name: "New Category"	The category should be removed from the list upon deletion.	Category "New Category" removed successfully.	

Post-Condition: The new category is successfully added to the database and is available in the category list.

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

In the context of software implementation, the introduction serves as a foundational overview of the project. It outlines the purpose of the application and its intended impact on users and organizational processes. This section highlights the goals of the implementation, which may include improving operational efficiency, enhancing user experience, and meeting specific business needs. It is essential to convey the significance of the software in streamlining workflows, reducing costs, and providing a competitive edge in the market. By setting the stage in this manner, stakeholders can better understand the context and importance of the implementation efforts.

6.2 IMPLEMENTATION PROCEDURES

Implementation procedures refer to the systematic steps taken to deploy the application software within an organization. This section details the various phases of the implementation process, including planning, installation, testing, and deployment. Each phase is critical to ensuring that the software functions as intended and meets the requirements set forth during the planning stage. Procedures may involve configuring the software to align with organizational policies, migrating existing data to the new system, and ensuring compatibility with current hardware. Additionally, this section emphasizes the importance of collaboration among different teams, including IT, management, and end-users, to facilitate a smooth transition and minimize disruptions to daily operations.

6.2.1 User Training

User training is a crucial component of the implementation process, aimed at equipping end-users with the knowledge and skills necessary to effectively utilize the new application software. This section outlines the training strategy, which may include hands-on workshops, online tutorials, and user manuals. Training should be tailored to different user roles within the organization, addressing their specific needs and use cases. Effective training fosters user confidence and proficiency, ensuring that employees can maximize the software’s capabilities. Furthermore, providing ongoing support and resources after initial training can help users adapt to any updates or changes to the software, ultimately leading to higher satisfaction and productivity.

6.2.2 Training on the Application Software

Training on the application software is a critical phase in the implementation process, designed to equip users with the necessary skills and knowledge to effectively operate the system. This training ensures that all users, regardless of their technical background, are comfortable navigating the software and utilizing its features. The program typically begins with a needs assessment to identify the specific requirements of different user groups, allowing for tailored training content. Comprehensive training materials, including user manuals and quick reference guides, are developed to support learning. Various delivery methods, such as in-person workshops and online courses, are employed to cater to diverse learning preferences. Practical exercises are integrated to provide hands-on experience, enabling users to apply their knowledge in real-world scenarios.

6.2.3 System Maintenance

System maintenance involves the ongoing support and updates required to ensure that the application software continues to operate smoothly and meets the evolving needs of the organization. This section covers various aspects of maintenance, including regular software updates, bug fixes, and performance monitoring. Establishing a robust maintenance plan is vital for identifying and addressing potential issues before they impact users.

6.2.4 Hosting

Hosting your project on Render allows you to deploy web applications without the hassle of managing server infrastructure. Render is a cloud platform that simplifies hosting by offering services like automatic scaling, continuous deployment, and easy integration with GitHub repositories.

Render Hosting

Render provides a platform for deploying web applications directly from a GitHub repository. For this project, **AgroRentHub**, the repository is hosted on GitHub, and the website is hosted on Render. The application uses **SQLite** as the database, which is suitable for small-to-medium applications. The website can be accessed through the link: https://agrorenthub.onrender.com/.

Procedure for Hosting a Website on Render

Step 1: Set up a Render Account

- Visit Render's website and sign up for a new account or log in if you already have one.

Step 2: Link GitHub Repository

- Once logged in to Render, navigate to the **Dashboard** and click on the **New** button to create a new service.
- Select **Web Service** as the type of service.
- When prompted, connect your **GitHub** account to Render if you haven't already done so.
- After linking your GitHub account, choose the **AgroRentHub** repository from the list of available repositories.

Step 3: Configure Deployment Settings

- Select the appropriate branch for deployment (usually main or master).
- Under **Environment**, choose the correct runtime for your project (for example, Python 3.x for Django applications).
- Set the **Build Command** to install dependencies: pip install -r requirements.txt.
- Set the **Start Command** to run the Django application: gunicorn AgroRentHub.wsgi.

Step 4: Set up the Database (SQLite)

- Since the project uses **SQLite** as the database, no complex database configuration is required on Render. However, make sure the database file is included in the repository and is properly configured within your project.
- If you plan to migrate your database, ensure that the correct Django commands are run during deployment to apply migrations (e.g., python manage.py migrate).

Step 5: Deploy the Project

- Once everything is set up, click **Create Web Service** to begin the deployment process.
- Render will automatically build the project, install dependencies, and start the web service.
- After the deployment is complete, your project will be live at the provided URL, e.g., https://agrorenthub.onrender.com/.

- **Hosted Website:** https://render.com/
- **Hosted Link:** https: https://agrorenthub.onrender.com/.
- **Hosted Link QR Code:**

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The AgroRent Hub project has successfully established a comprehensive online platform for renting agricultural equipment and services, bridging the gap between farmers and suppliers. This innovative solution not only streamlines the rental process but also enhances accessibility to essential resources for agricultural productivity. By integrating features such as user-friendly navigation, secure payment gateways, and robust support for delivery management, AgroRent Hub has created a responsive ecosystem that meets the diverse needs of its users. The project has demonstrated its potential to revolutionize the agricultural sector, empowering farmers to optimize their operations while fostering a community of collaboration and efficiency. The implementation of a training program ensures that users are well-equipped to leverage the platform effectively, promoting sustainable agricultural practices. As the AgroRent Hub continues to evolve, the commitment to user satisfaction and operational excellence remains at the forefront of its objectives.

7.2 FUTURE SCOPE

The future scope of the AgroRent Hub project is promising, with several opportunities for expansion and enhancement. First, the platform can integrate advanced technologies, such as artificial intelligence and machine learning, to offer personalized recommendations based on user behavior and preferences. This could significantly improve the user experience and drive higher engagement levels. Second, expanding the range of rental products and services to include specialized equipment and maintenance services would cater to a broader audience, ultimately increasing the platform's market reach. Additionally, implementing a mobile application would allow users to access the platform conveniently, enabling them to manage their rentals on-the-go.

Furthermore, AgroRent Hub could explore partnerships with agricultural research institutions to provide users with valuable insights and data-driven recommendations on best practices and equipment usage. This collaboration would enhance the platform's value proposition and position it as a thought leader in the agricultural rental space. Finally, expanding to international markets could open new avenues for growth, allowing AgroRent Hub to cater to a global audience and contribute to sustainable agricultural practices worldwide. By embracing these opportunities, AgroRent Hub can solidify its role as a transformative force in the agriculture sector, ultimately leading to increased productivity and profitability for farmers and suppliers alike.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

1. S. Johari, A. F. Yassin, and S. Q. Abdellah, "Metal Artifact Suppression in Dental Cone Beam Computed Tomography Images Using Image Processing Techniques," International Journal ofAdvanced Computer Science and Applications, vol. 11, no. 9, pp. 131–137, 2020.

2. R. Gulo, C. A. M. Remya, and T. S. Ramanathan, "Techniques of Medical Image Processing and Analysis Accelerated by High-Performance Computing," Biomedical Research, vol. 31, no. 2.

3. M. Almeida, A. Ribeiro, and R. V. Gonçalves, "Image Processing as a Tool for EvaluatingDenture Adhesives Removal Techniques," Journal of Dentistry.

4. S. Georgis, H. Li, and D. Zhang, "Acceleration Techniques and Evaluation on Multi-Core CPU, GPU, and FPGA for Image Processing and Super-Resolution," IEEE Transactions on Parallel andDistributed Systems, vol. 31, no. 8, pp. 1831–1842, 2020.

5. Y. Wei, Z. Zhang, and L. Liu, "Railway Track Fastener Defect Detection Based on Image Processing and Deep Learning Techniques," IEEE Access, vol. 8, pp. 174–183, 2020.

6. K. Sun, Q. Zhang, and X. Yang, "Precision Agriculture through UAV-based Hyperspectral Imaging," Journal of Agricultural and Food Chemistry, vol. 68, no. 32, pp. 12814–12823, 2020.

7. M. R. Choi, J. Lee, and H. Y. Kim, "Application of Multispectral Imaging Techniques in PrecisionAgriculture: A Review," Journal of Agricultural Science and Technology, vol. 22, no. 3, pp. 99– 108, 2020.

8. G. S. Ajith, K. S. Mohan, and S. Ramachandran, "Recent Trends in Agricultural Robotics and Automation: A Comprehensive Review," Journal of Field Robotics, vol. 37, no. 4, pp. 536–556,2020.

9. W. Thomas, and R. P. Singh, "AI-Driven Image Processing Techniques in Agricultural EquipmentManagement: A Case Study," IEEE Transactions on Automation Science and Engineering, vol. 18, no. 3, pp. 977–990, 2020.

10. N. Kumar, R. Arora, and S. Singh, "Challenges and Opportunities in AI-Based EquipmentManagement in Agriculture," Computers and Electronics in Agriculture, vol. 178, no. 2, p.105703, 2020.

11. P. Patil, M. Kumar, and H. Choudhary, "Role of Image Processing in Agriculture: An Overview,"Computers in Agriculture, vol. 45, no. 3, pp. 67.

12. D. S. Chouhan, R. Pradhan, and M. G. Reddy, "Image Processing Techniques for Agricultural Equipment Identification," Image and Vision Computing, vol. 98, p. 103934, 2020.

13. S. Vijayan, R. K. Srivastava, and B. R. Bhowmik, "Artificial Intelligence in Agriculture: AReview of the State of the Art," Expert Systems with Applications, vol. 173.

14. T. L. Wang, Y. K. Yu, and C. Z. Shen, "Integrating Image Processing and AI for Agricultural Equipment Management," Future Generation Computer Systems, vol. 108,

pp. 170–184, 2020.

15. J. P. Gupta and P. K. Srivastava, "A Comprehensive Review on AI-Based Techniques forPrecision Agriculture," Artificial Intelligence in Agriculture, vol. 3, pp. 1–14, 2020.

16. M. P. Singh, and R. P. Sharma, "Challenges and Opportunities of AI and Image Processing inAgriculture," Smart Agriculture, vol. 6, no. 2, pp. 89–97, 2020.

17. P. K. Agarwal, S. K. Misra, and M. K. Jain, "Recent advances in image processing for agriculturalapplications," Information Processing in Agriculture, vol. 7, no. 3, pp. 300–310.

18. R. K. Singh, and S. Gupta, "Opportunities and Challenges of Artificial Intelligence in AgriculturalEquipment Management," Agricultural Systems, vol. 184, p. 102918, 2020.

19. D. P. Goyal, A. S. Mathur, and N. K. Reddy, "AI-Driven Image Processing Techniques inAgriculture: A Comprehensive Review," IEEE Transactions on Automation Science and Engineering, vol. 17, no. 4, pp. 1887.

20. AgriHub. (n.d.). AgriHub: Connecting Farmers and SuppliersRetrieved from https://www.agrihub.com

21. National Agriculture Statistics Service. (2024). Agricultural Equipment and Machinery in theUnited States. Retrieved from https://www.nass.usda.gov

22. World Bank. (2024). Agricultural Productivity and TechnologyRetrieved fromhttps://www.worldbank.org

23. American Society of Agricultural and Biological Engineers (ASABE). (2024). Advancementsin Agricultural Technology. Retrieved from https://www.asabe.org

24. Farm Equipment. (2024). Trends in Agricultural Equipment Rentals. Retrieved from https://www.farmequipment.com

25. The Agriculture and Food Security Center. (2024). Impact of Technology on Agriculture.Retrieved from https://www.ifsan.org

26. AgFunder Network Partners. (2024). Investment in AgriTech: Trends and Opportunities.Retrieved from https://www.agfunder.com

27. Global Agriculture Information Network (GAIN). (2024). Market and Trade Data for Agriculture. Retrieved from https://www.fas.usda.gov/gain

CHAPTER 9

APPENDIX

9.1 Sample Code

Cart Functionality

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cart</title>
    {% load static %}
    {% load custom_filters %}
    <link rel="stylesheet" href="{% static 'css/customer/style.css' %}">
    <script src="{% static 'js/customer/script.js' %}" defer></script>
      <!-- Include SweetAlert2 CSS -->
      <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/sweetalert2@11/dist/sweetalert2.min.css">
      <script src="{% static 'js/customer/script.js' %}" defer></script>
      <!-- Include SweetAlert2 JavaScript -->
      <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
</head>
<body>
    {% if request.session.user_id %}
    <header>
        <div class="logo">AgroRentHub</div>
        <nav>
            <ul>
                <li><a href="{% url 'customer_index' %}">Home</a></li>
                <li><a href="{% url 'customer_service' %}">Services</a></li>
                <li><a href="{% url 'products' %}">Products</a></li>
                <li><a href="{% url 'productcart' %}">Cart</a></li>
                <li><a href="{% url 'orderdetails' %}">Orders</a></li>
                <li><a href="{% url 'customer_profile' %}">Profile</a></li>
                <li><a href="{% url 'logout' %}">Logout</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section class="content">
            <div  class="welcome-container">
                <img class="welcome-image" src="{% static 'images/welcome.jpg' %}" alt="Welcome
Image">
             <div class="welcome-message">
                <h1>Rental Product Cart</h1>
                </div>
            </div>
           <!-- {% if messages %}
            <ul class="messages">
                {% for message in messages %}
                <li{% if message.tags %} class="{{ message.tags }}"{% endif %}>{{ message }}</li>
                {% endfor %}
            </ul>
            {% endif %}-->
            <div  class="table-summary-container">
                <div class="table-container">
                {% if cart_items %}
                <table>
                    <thead>
                        <tr>
                            <th>Si.No</th>
                            <th>Product Image</th>
                            <th>Price Per Day</th>
                            <th>Quantity</th>
                            <th>Start Date</th>
                            <th>End Date</th>
                            <th>Days</th>
                            <th>Price</th>
                            <th>Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for item in cart_items %}
                        <tr>
                            <td class="si-no">{{ forloop.counter }}</td>
                            <td>
                                {% if item.product %}
                                <a href="{% url 'product_detail' item.product.id %}">
                                    <img src="{{ item.product.product_image.url }}" alt="{{
item.product.product_name }}">
                                </a><br>
                                <br>{{ item.product.product_name }}
                                {% else %}
                                <p>Product not available</p>
                                {% endif %}
                            </td>
                            <td>₹{{ item.product.product_price_per_day }}</td>
                            <td>{{ item.quantity }}</td>
                            <td>{{ item.start_date }}</td>
                            <td>{{ item.end_date }}</td>
                            <td>{{  item.start_date|date_difference:item.end_date  }}</td>
                            <td>₹{{ item|total_price }}</td>
                            <td>
                                <form action="{% url 'delete_cart_item' item.id %}" method="post">
                                    {% csrf_token %}
                                    <button type="submit" class="delete-button">Delete</button>
                                </form>
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
                {% else %}
                <div  class="empty-cart-message">
                    <img class="empty-cart-icon" src="{% static 'images/empty_cart.gif' %}"
alt="Empty Cart Icon">
                    <h2>Your cart is empty.</h2>
                </div>
                {% endif %}
                </div>
                {% if cart_items %}
                <div class="summary-container">
                {% for item in cart_items %}
```

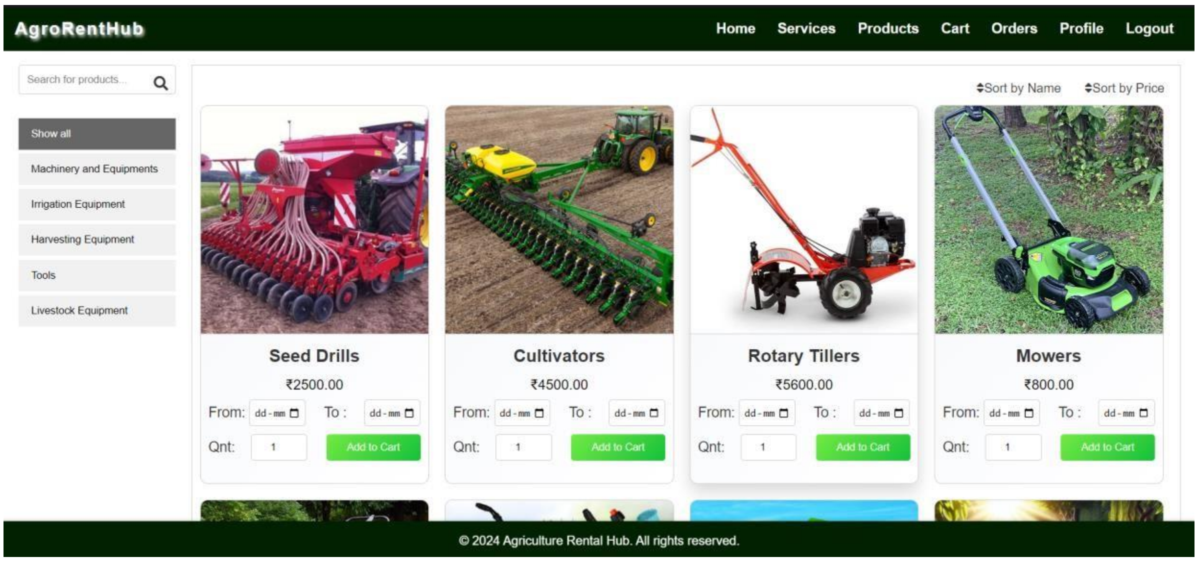
```
                    <div class="summary-item">
                         <span>{{ item.product.product_name }} ({{ item.quantity }} x
₹{{item.product.product_price_per_day }} x {{ item.start_date|date_difference:item.end_date
}}days)</span>
                        <span>₹{{ item|total_price }}</span>
                    </div>
                    {% endfor %}
                    <div class="summary-item summary-total">
                        <span>Total:</span>
                        <span>₹{{ cart_items|sum_prices }}</span>
                    </div>
                    <form action="{% url 'place_order' %}" method="post">
                        {% csrf_token %}
                        <button type="submit" class="place-order-button">Place Order</button>
                    </form>
                </div>
                {% endif %}
            </div>
        </section>
        <!-- SweetAlert for Django messages -->
{% if messages %}
<script>
    document.addEventListener("DOMContentLoaded", function() {
        {% for message in messages %}
        // Mapping Django message tags to SweetAlert icons
        let iconMap = {
            'debug': 'info',
            'info': 'info',
            'success': 'success',
            'warning': 'warning',
            'error': 'error',
        };
        // Get the appropriate icon for SweetAlert
        let icon = iconMap['{{ message.tags }}'] || 'info'; // Default to 'info' if tag isn't found
        Swal.fire({
            title: '{{ message.tags|title }}',
            text: '{{ message }}',
            icon: icon,
            confirmButtonText: 'OK'
        });
        {% endfor %}
    });
</script>
{% endif %}
    </main>
    <footer class="footer">
        <p>&copy; 2024 Agriculture Rental Hub. All rights reserved.</p>
    </footer>
    {% else %}
    <p>Access Denied</p>
    {% endif %}
    <script src="{% static 'js/customer/script.js' %}" defer></script>
</body>
</html>
```

Views.py

```
def add_to_cart(request, product_id):
    user_id = request.session.get("user_id")
    user = get_object_or_404(Farmer, id=user_id)
    product = get_object_or_404(Product, id=product_id)
    quantity = int(request.POST.get("quantity", 1))
    start_date_str = request.POST.get("start_date", None)
    end_date_str = request.POST.get("end_date", None)
    if start_date_str:
        start_date = datetime.strptime(start_date_str, "%Y-%m-%d").date()
    else:
        start_date = None
    if end_date_str:
        end_date = datetime.strptime(end_date_str, "%Y-%m-%d").date()
    else:
        end_date = None
    if not start_date or not end_date or end_date <= start_date:
        messages.error(request, "Invalid start or end date.")
        return redirect("product_detail", product_id=product_id)
    if product.stock_quantity < quantity:
        messages.error(request, "Not enough stock available.")
        return redirect("product_detail", product_id=product_id)
    if product.stock_quantity <= 5:
        messages.error(request, "Product is out of stock.")
        return redirect("product_detail", product_id=product_id)
    number_of_days = (end_date - start_date).days
    price = product.product_price_per_day * quantity * number_of_days
    cart_item, created = CartItem.objects.get_or_create(
        user=user,
        product=product,
        defaults={
            "quantity": quantity,
            "start_date": start_date,
            "end_date": end_date,
            "price": price,
        },
    )
    if not created:
        cart_item.quantity = quantity
        cart_item.price = (
            product.product_price_per_day * cart_item.quantity * number_of_days
        )
        cart_item.start_date = start_date
        cart_item.end_date = end_date
        cart_item.save()
    return redirect("productcart")
def productcart(request):
    user_id = request.session.get("user_id")
    if not user_id:
        return redirect("login")  # Redirect to login if no user_id in session
    user = get_object_or_404(Farmer, id=user_id)
    cart_items = CartItem.objects.filter(user=user)
    return render(request, "customer/productcart.html", {"cart_items": cart_items})
def delete_cart_item(request, item_id):
    cart_item = get_object_or_404(CartItem, id=item_id)
    cart_item.delete()
```

```
return redirect("productcart")
```

9.1 Screen Shots



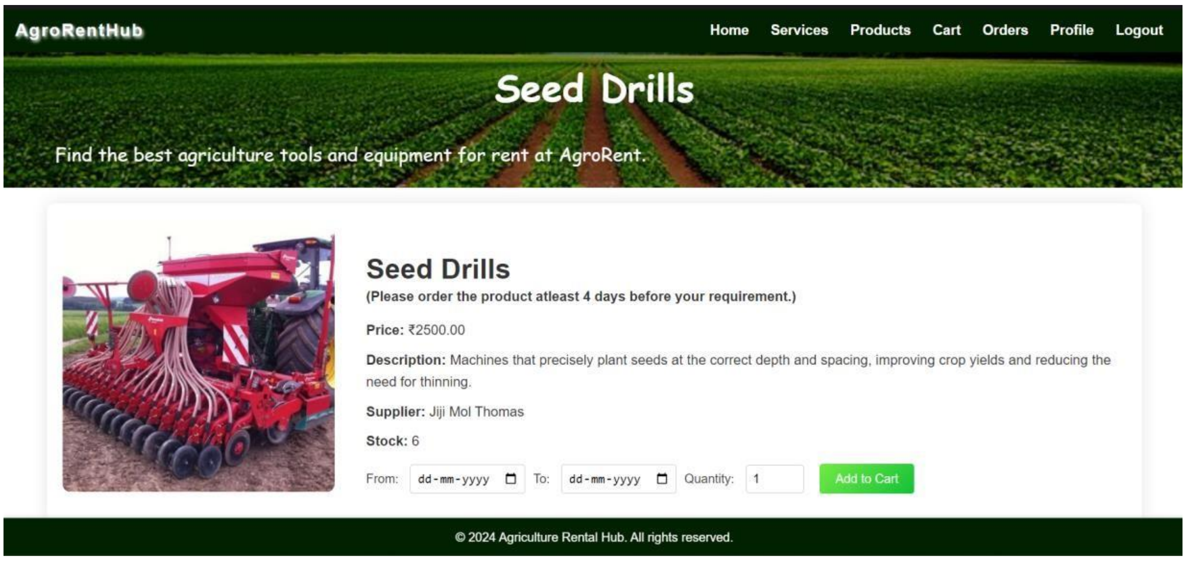
Seed Drills ₹2500.00

Cultivators ₹4500.00

Rotary Tillers ₹5600.00

Mowers ₹800.00

© 2024 Agriculture Rental Hub. All rights reserved.



Seed Drills

(Please order the product atleast 4 days before your requirement.)

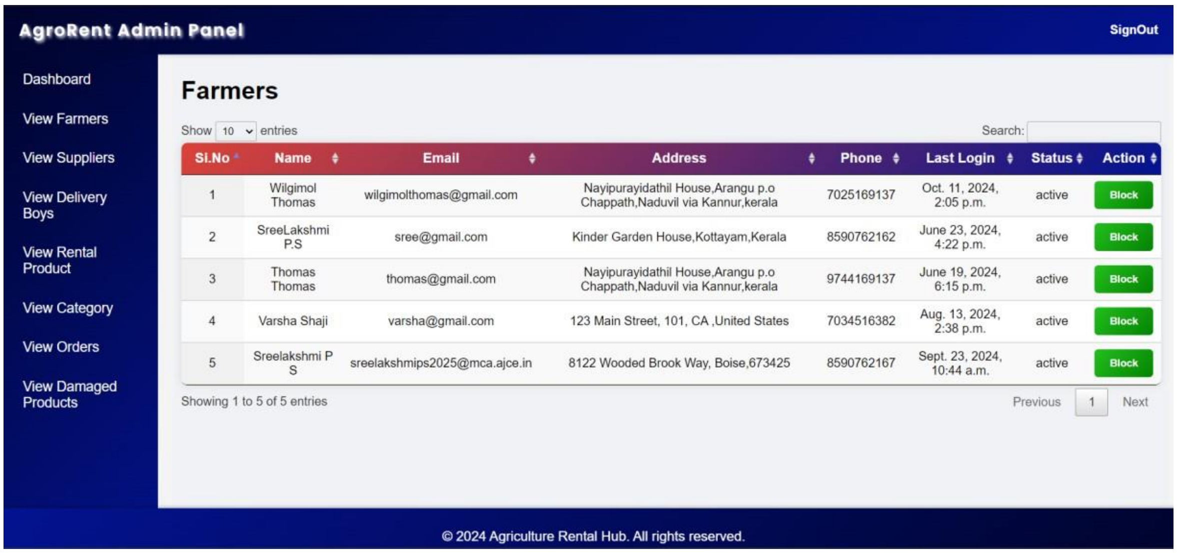
Price: ₹2500.00

Description: Machines that precisely plant seeds at the correct depth and spacing, improving crop yields and reducing the need for thinning.

Supplier: Jiji Mol Thomas

Stock: 6

© 2024 Agriculture Rental Hub. All rights reserved.

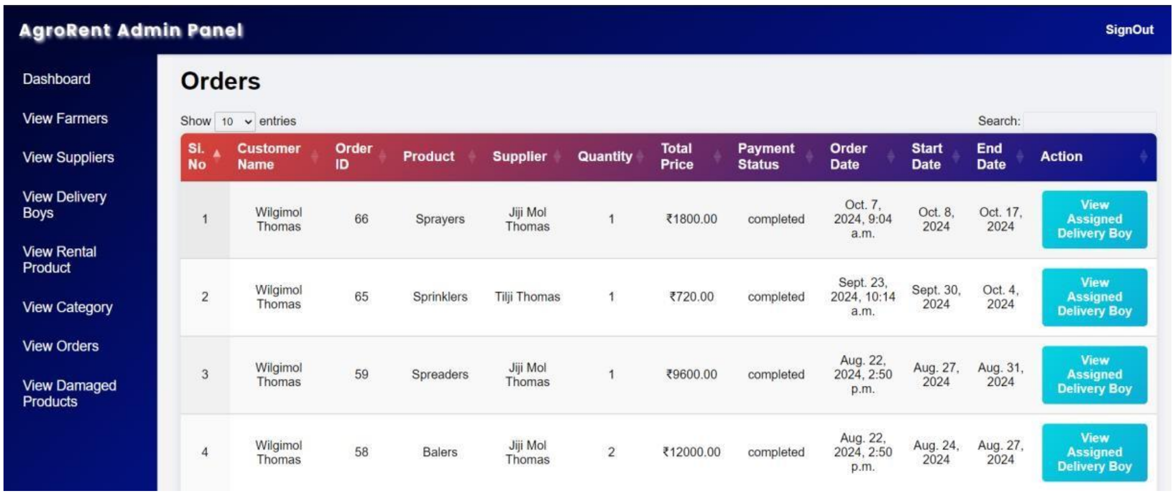


Farmers

Sl.No	Name	Email	Address	Phone	Last Login	Status	Action
1	Wilgimol Thomas	wilgimolthomas@gmail.com	Nayipurayidathil House,Arangu p.o Chappath,Naduvil via Kannur,kerala	7025169137	Oct. 11, 2024, 2:05 p.m.	active	Block
2	SreeLakshmi P.S	sree@gmail.com	Kinder Garden House,Kottayam,Kerala	8590762162	June 23, 2024, 4:22 p.m.	active	Block
3	Thomas Thomas	thomas@gmail.com	Nayipurayidathil House,Arangu p.o Chappath,Naduvil via Kannur,kerala	9744169137	June 19, 2024, 6:15 p.m.	active	Block
4	Varsha Shaji	varsha@gmail.com	123 Main Street, 101, CA ,United States	7034516382	Aug. 13, 2024, 2:38 p.m.	active	Block
5	Sreelakshmi P S	sreelakshmips2025@mca.ajce.in	8122 Wooded Brook Way, Boise,673425	8590762167	Sept. 23, 2024, 10:44 a.m.	active	Block

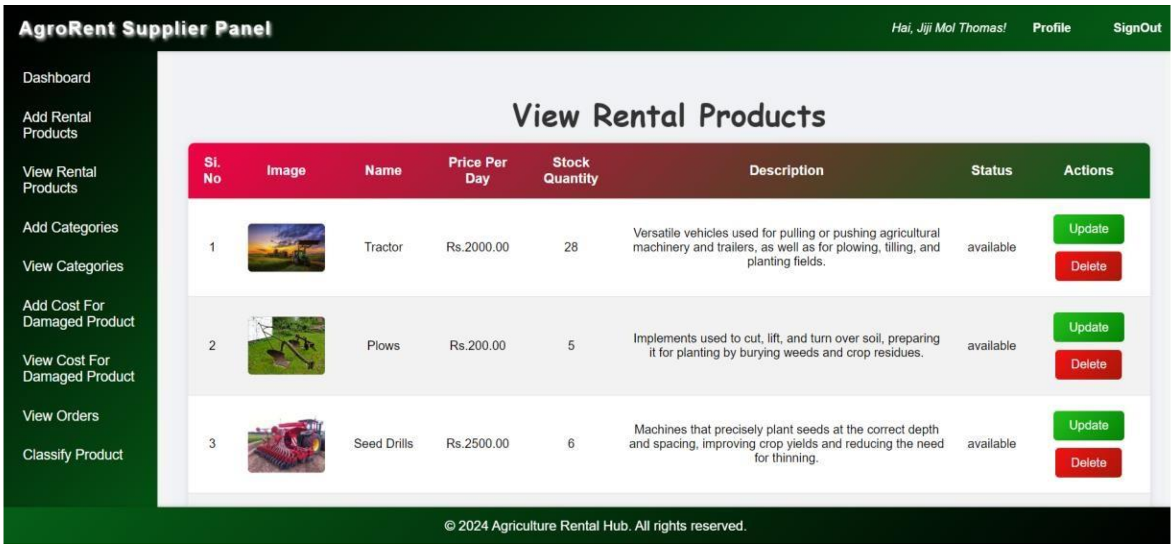
Showing 1 to 5 of 5 entries

© 2024 Agriculture Rental Hub. All rights reserved.



Orders

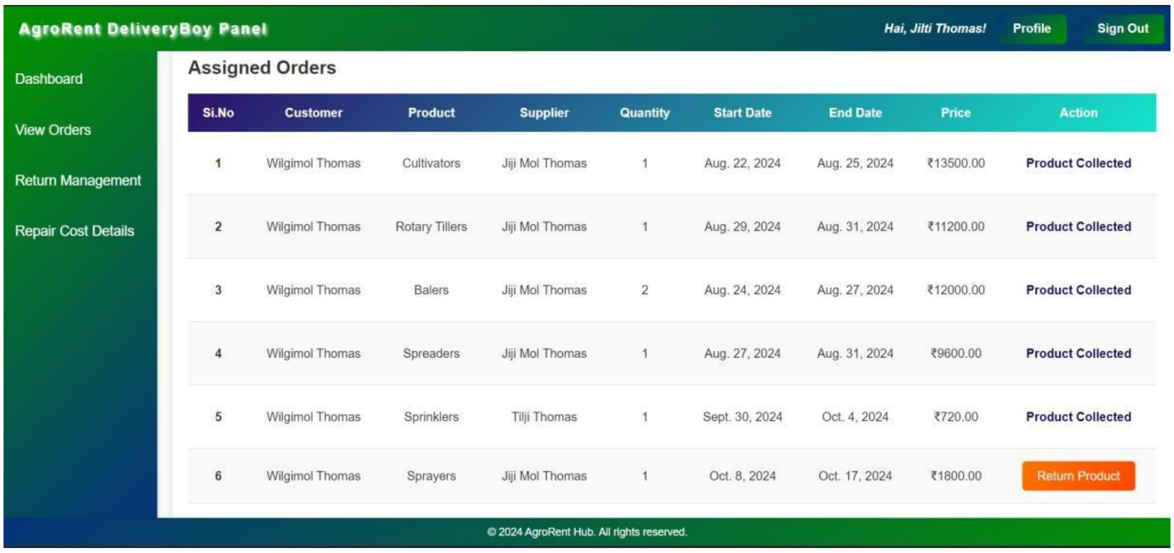
Sl. No	Customer Name	Order ID	Product	Supplier	Quantity	Total Price	Payment Status	Order Date	Start Date	End Date	Action
1	Wilgimol Thomas	66	Sprayers	Jiji Mol Thomas	1	₹1800.00	completed	Oct. 7, 2024, 9:04 a.m.	Oct. 8, 2024	Oct. 17, 2024	View Assigned Delivery Boy
2	Wilgimol Thomas	65	Sprinklers	Tilji Thomas	1	₹720.00	completed	Sept. 23, 2024, 10:14 a.m.	Sept. 30, 2024	Oct. 4, 2024	View Assigned Delivery Boy
3	Wilgimol Thomas	59	Spreaders	Jiji Mol Thomas	1	₹9600.00	completed	Aug. 22, 2024, 2:50 p.m.	Aug. 27, 2024	Aug. 31, 2024	View Assigned Delivery Boy
4	Wilgimol Thomas	58	Balers	Jiji Mol Thomas	2	₹12000.00	completed	Aug. 22, 2024, 2:50 p.m.	Aug. 24, 2024	Aug. 27, 2024	View Assigned Delivery Boy



View Rental Products

Sl. No	Image	Name	Price Per Day	Stock Quantity	Description	Status	Actions
1		Tractor	Rs.2000.00	28	Versatile vehicles used for pulling or pushing agricultural machinery and trailers, as well as for plowing, tilling, and planting fields.	available	Update Delete
2		Plows	Rs.200.00	5	Implements used to cut, lift, and turn over soil, preparing it for planting by burying weeds and crop residues.	available	Update Delete
3		Seed Drills	Rs.2500.00	6	Machines that precisely plant seeds at the correct depth and spacing, improving crop yields and reducing the need for thinning.	available	Update Delete

© 2024 Agriculture Rental Hub. All rights reserved.

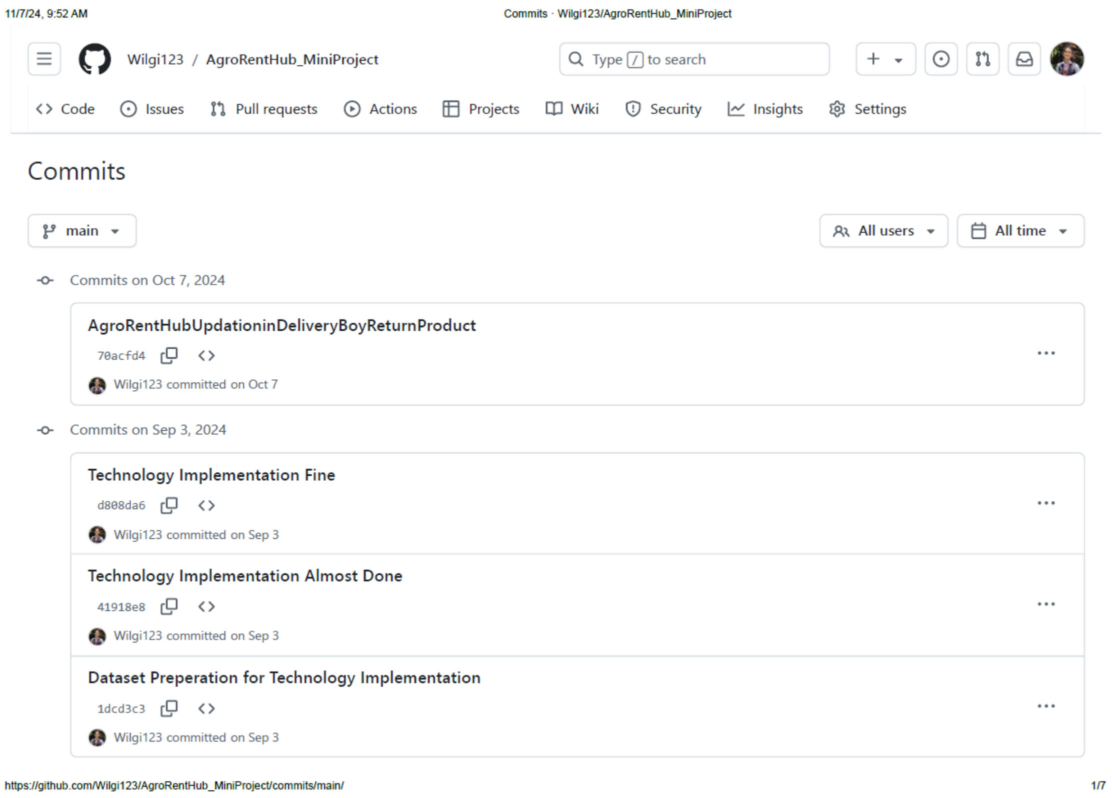


Assigned Orders

Sl.No	Customer	Product	Supplier	Quantity	Start Date	End Date	Price	Action
1	Wilgimol Thomas	Cultivators	Jiji Mol Thomas	1	Aug. 22, 2024	Aug. 25, 2024	₹13500.00	Product Collected
2	Wilgimol Thomas	Rotary Tillers	Jiji Mol Thomas	1	Aug. 29, 2024	Aug. 31, 2024	₹11200.00	Product Collected
3	Wilgimol Thomas	Balers	Jiji Mol Thomas	2	Aug. 24, 2024	Aug. 27, 2024	₹12000.00	Product Collected
4	Wilgimol Thomas	Spreaders	Jiji Mol Thomas	1	Aug. 27, 2024	Aug. 31, 2024	₹9600.00	Product Collected
5	Wilgimol Thomas	Sprinklers	Tilji Thomas	1	Sept. 30, 2024	Oct. 4, 2024	₹720.00	Product Collected
6	Wilgimol Thomas	Sprayers	Jiji Mol Thomas	1	Oct. 8, 2024	Oct. 17, 2024	₹1800.00	Return Product

© 2024 AgroRent Hub. All rights reserved.

9.2 GIT LOG



11/7/24, 9:52 AM Commits · Wilgi123/AgroRentHub_MiniProject

Commits

main · All users · All time

Commits on Oct 7, 2024

AgroRentHubUpdationinDeliveryBoyReturnProduct
70acfd4
Wilgi123 committed on Oct 7

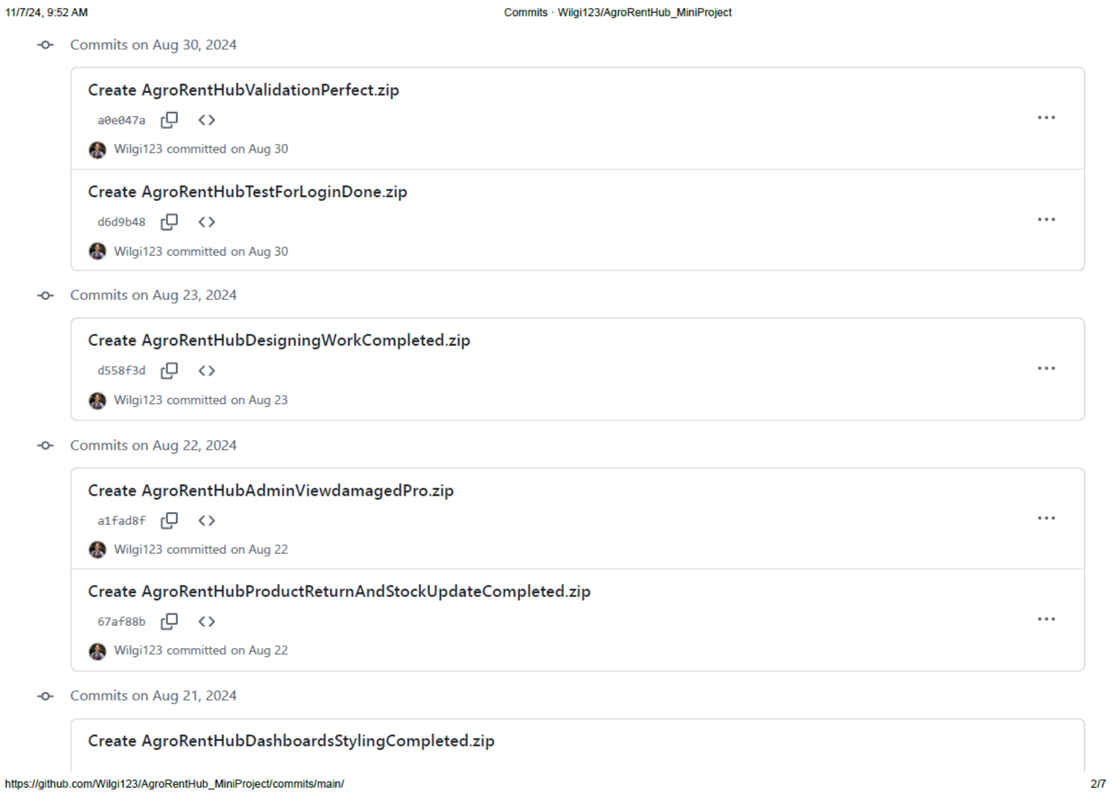
Commits on Sep 3, 2024

Technology Implementation Fine
d808da6
Wilgi123 committed on Sep 3

Technology Implementation Almost Done
41918e8
Wilgi123 committed on Sep 3

Dataset Preperation for Technology Implementation
1dcd3c3
Wilgi123 committed on Sep 3

https://github.com/Wilgi123/AgroRentHub_MiniProject/commits/main/ 1/7



11/7/24, 9:52 AM Commits · Wilgi123/AgroRentHub_MiniProject

Commits on Aug 30, 2024

Create AgroRentHubValidationPerfect.zip
a0e047a
Wilgi123 committed on Aug 30

Create AgroRentHubTestForLoginDone.zip
d6d9b48
Wilgi123 committed on Aug 30

Commits on Aug 23, 2024

Create AgroRentHubDesigningWorkCompleted.zip
d558f3d
Wilgi123 committed on Aug 23

Commits on Aug 22, 2024

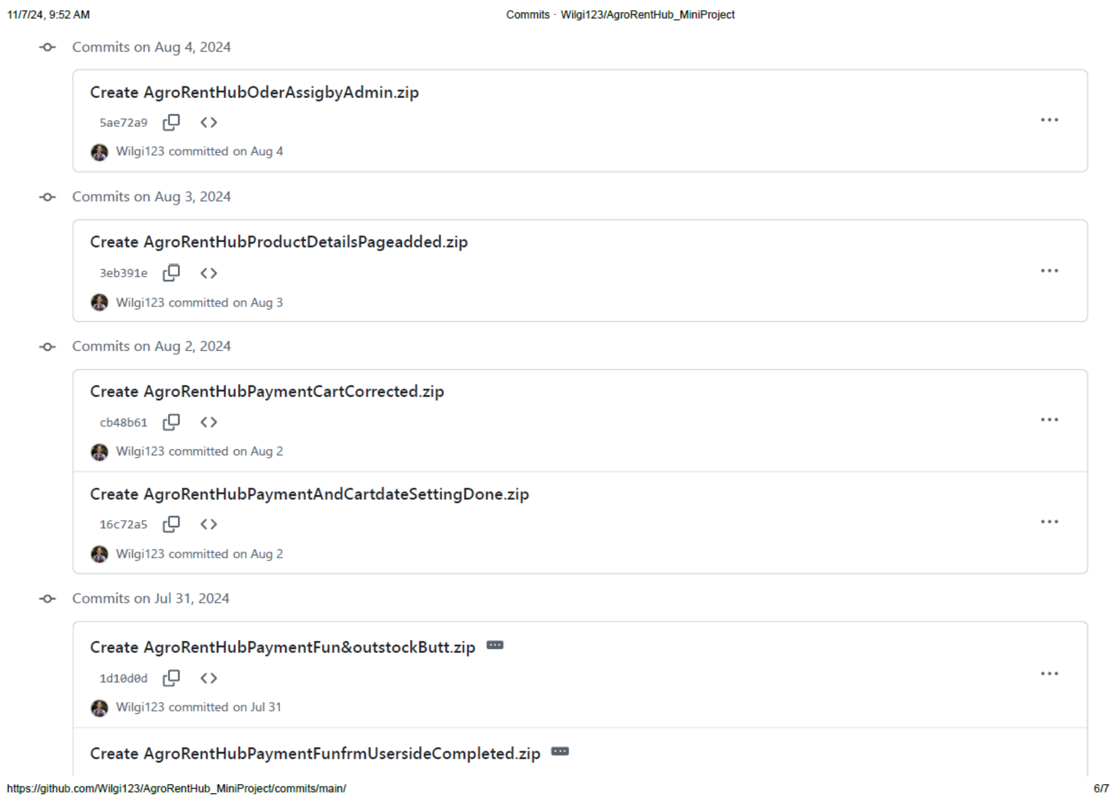
Create AgroRentHubAdminViewdamagedPro.zip
a1fad8f
Wilgi123 committed on Aug 22

Create AgroRentHubProductReturnAndStockUpdateCompleted.zip
67af88b
Wilgi123 committed on Aug 22

Commits on Aug 21, 2024

Create AgroRentHubDashboardsStylingCompleted.zip

https://github.com/Wilgi123/AgroRentHub_MiniProject/commits/main/ 2/7



11/7/24, 9:52 AM Commits · Wilgi123/AgroRentHub_MiniProject

Commits on Aug 4, 2024

Create AgroRentHubOderAssigbyAdmin.zip
5ae72a9
Wilgi123 committed on Aug 4

Commits on Aug 3, 2024

Create AgroRentHubProductDetailsPageadded.zip
3eb391e
Wilgi123 committed on Aug 3

Commits on Aug 2, 2024

Create AgroRentHubPaymentCartCorrected.zip
cb48b61
Wilgi123 committed on Aug 2

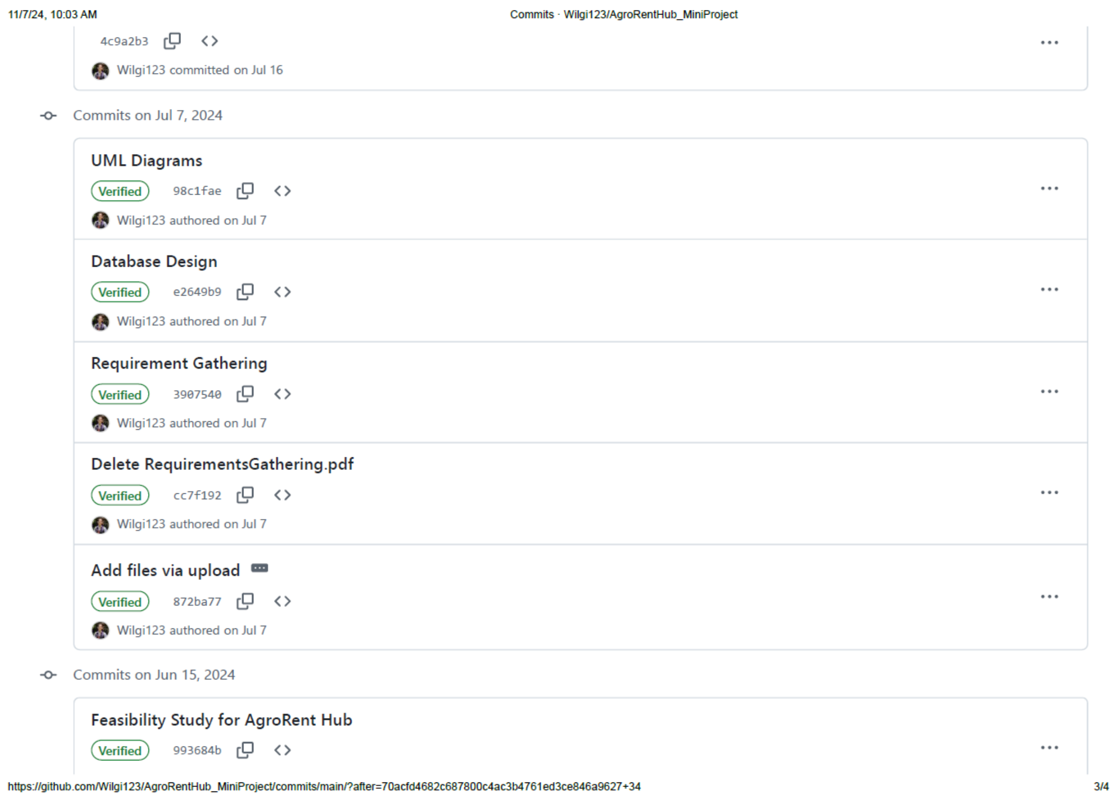
Create AgroRentHubPaymentAndCartdateSettingDone.zip
16c72a5
Wilgi123 committed on Aug 2

Commits on Jul 31, 2024

Create AgroRentHubPaymentFun&outstockButt.zip
1d10d0d
Wilgi123 committed on Jul 31

Create AgroRentHubPaymentFunfrmUsersideCompleted.zip

https://github.com/Wilgi123/AgroRentHub_MiniProject/commits/main/ 6/7



11/7/24, 10:03 AM Commits · Wilgi123/AgroRentHub_MiniProject

4c9a2b3
Wilgi123 committed on Jul 16

Commits on Jul 7, 2024

UML Diagrams
Verified 98c1fae
Wilgi123 authored on Jul 7

Database Design
Verified e2649b9
Wilgi123 authored on Jul 7

Requirement Gathering
Verified 3907540
Wilgi123 authored on Jul 7

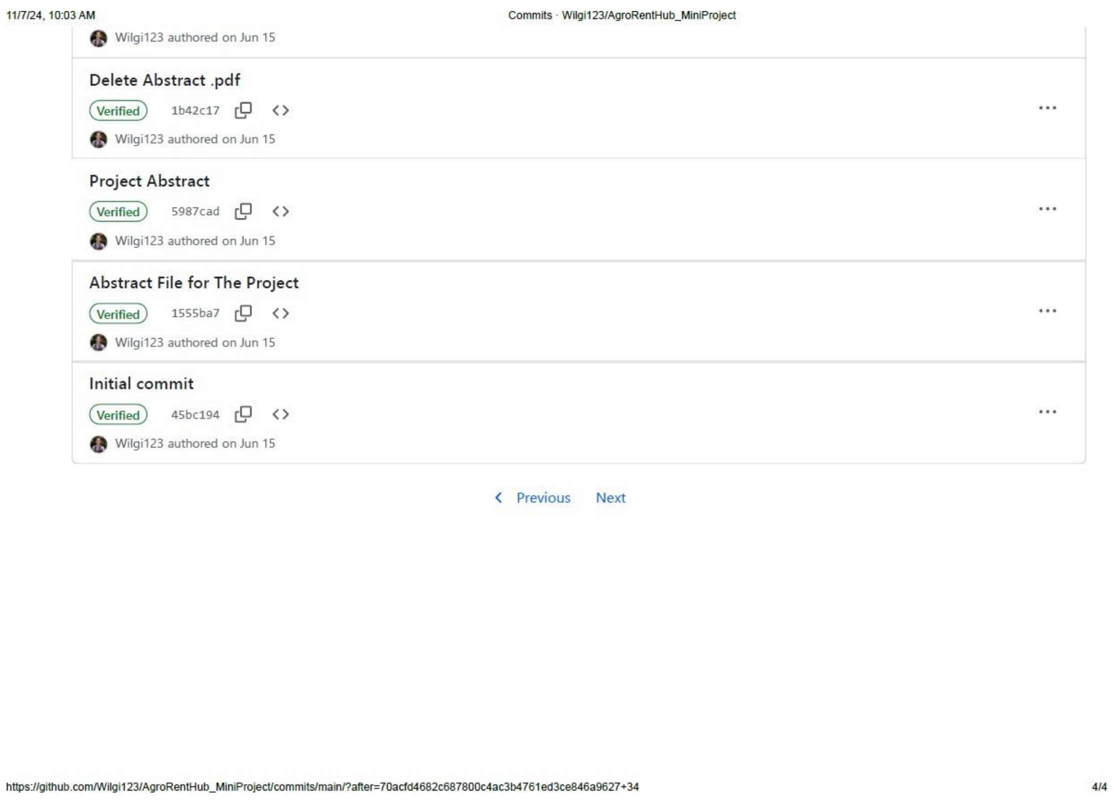
Delete RequirementsGathering.pdf
Verified cc7f192
Wilgi123 authored on Jul 7

Add files via upload
Verified 872ba77
Wilgi123 authored on Jul 7

Commits on Jun 15, 2024

Feasibility Study for AgroRent Hub
Verified 993684b

https://github.com/Wilgi123/AgroRentHub_MiniProject/commits/main/?after=70acfd4682c687800c4ac3b4761ed3ce846a9627+34 3/4



11/7/24, 10:03 AM Commits · Wilgi123/AgroRentHub_MiniProject

Wilgi123 authored on Jun 15

Delete Abstract .pdf
Verified 1b42c17
Wilgi123 authored on Jun 15

Project Abstract
Verified 5987cad
Wilgi123 authored on Jun 15

Abstract File for The Project
Verified 1555ba7
Wilgi123 authored on Jun 15

Initial commit
Verified 45bc194
Wilgi123 authored on Jun 15

< Previous Next

https://github.com/Wilgi123/AgroRentHub_MiniProject/commits/main/?after=70acfd4682c687800c4ac3b4761ed3ce846a9627+34 4/4