# Report 3: Core Feature Development Report (Development Phase)

## Development Progress Overview

* The development of the equipment rental management system is on track. The core backend logic and most user interfaces have been implemented.

**Completed**:

* Equipment management
* Rental contract creation
* Rental details and return handling
* Inventory stock update
* Role-based login

**Pending**:

* Notification system
* Full CI/CD integration
* Advanced reporting/statistics

## Implemented Features

 **Equipment Management**: CRUD operations on rental equipment, status tracking (Available, Rented, Under Maintenance).

 **Rental Contract Creation**: Allows staff to create new contracts, select equipment, and generate rental details.

 **Return Processing**: Enables staff to mark equipment as returned and calculate fees.

 **Inventory Sync**: Automatically updates quantity in stock and equipment availability after each return.

 **Authentication & Role Management**: Basic login system to distinguish Owner and Staff role

## Technical Implementation

 **OOP Principles**: The project applies encapsulation (via private fields and public properties), inheritance (base service layers)

 **EF Core**: Used for database interaction via DbContext, with code-first approach and migrations.

 **WPF**:

* XAML used for responsive and modern UI
* Data binding enables reactive UIs

## Challenges and Solutions

|  |  |
| --- | --- |
| **Level** | **Mitigation measures** |
| |  | | --- | | RentalDetails disappearing on return |  |  | | --- | |  | | |  | | --- | | Fixed issue by checking EF EntityState.Deleted during session save |  |  | | --- | |  | |
| |  | | --- | | Inventory not updating correctly |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Refactored logic to update quantities in the return method |  |  | | --- | |  | |  |  | | --- | |  | |

## Git Commit History

* <https://github.com/Lunatik137/EquipmentRentalManagementApp>

## Code Quality and Documentation

 **Naming Conventions**: CamelCase for variables, PascalCase for classes and methods.

 **Level Structure**: Clean separation of Data Access, Repository, Service, App.

 **Code Comments**: Provided in all service and controller classes.

## Testing Activities

* Using Unit Test By Console App

## Next Steps

 Integrate full combo rental and promotion support

 Implement notification system for overdue rentals

 Add role-based dashboard summaries

 Finalize CI/CD pipeline and deploy to test environment

 Perform end-to-end testing and bug fixes