UNIVERSITY of TECHNOLOGY & ENTREPRENEURSHIP

Field	Student Input
Project Title	Vehicle Inventory and Sales Management System for SMEs
Team Members	Lay Katsrieng Sour Chansokpanha Heang Sovannara
Course/ Class	Advanced OOP with Python / Class CSM1
Lecturer	Sek Socheat
Semester / Year	Term 5 / Year 2
Submission Type	Proposal

University of Technology and Entrepreneurship – UTE

Abstract

Small and Medium Enterprises (SMEs) that sell vehicles need an effective way to manage vehicle inventory and sales records. Some businesses may use manual records, spreadsheets, or separate files to manage their daily operations. This can make it difficult to track available and sold vehicles, record stock movements, manage sales, and retrieve important information.

This project proposes the development of a web-based Vehicle Inventory and Sales Management System for SMEs. The system will help users manage vehicle inventory, stock movements, customers, sales, and invoices in one centralized system. It will also include Role-Based Access Control (RBAC) to control user access based on their roles.

The system will be developed using Python and Flask, with MySQL as the database. Object-Oriented Programming principles will be applied by organizing the system into domain models, services, repositories, and routes. The expected result is a functional and well-organized web-based system that helps SMEs manage vehicle inventory and sales operations more efficiently.

1. Introduction

Small and Medium Enterprises play an important role in business and economic activities. For SMEs involved in vehicle sales, managing vehicle inventory and sales information is an important part of daily operations. Businesses need accurate information about available vehicles, sold vehicles, sales transactions, customers, and invoices.

A proper management system can help organize this information and make it easier for authorized users to access and manage business records. Therefore, this project proposes **a web-based Vehicle Inventory and Sales Management System designed for Small and Medium Enterprises.**

The system will focus on vehicle inventory management, stock movement tracking, sales management, invoice management, and user access control. It will be developed using **Python, Flask, MySQL, and Object-Oriented Programming principles.**

2. Problem Statement

Many SMEs may manage vehicle inventory and sales records using manual methods, spreadsheets, or separate files. These methods can make it difficult to maintain accurate and updated information. Staff may need to check different records to determine whether a vehicle is available, sold, or recently added to the inventory.

Manual processes can also cause errors when recording sales and stock movements. It may be difficult to track the history of a vehicle, retrieve previous sales records, and manage invoices. In addition, without proper access control, unauthorized users may be able to access or modify important business information.

Therefore, a centralized web-based system is needed to manage vehicle inventory and sales operations. The proposed system will allow authorized users to manage vehicle records, track stock movements, process sales, generate invoices, and access reports based on their assigned roles.

3. Objectives

3.1 General Objective

To design and develop a web-based Vehicle Inventory and Sales Management System for Small and Medium Enterprises (SMEs) using Python, Flask, MySQL, and Object-Oriented Programming principles.

3.2 Specific Objectives

1. To develop a vehicle inventory management module that allows authorized users to create, view, update, and manage vehicle records.
2. To implement a stock movement tracking module to record vehicle additions, sales, and inventory adjustments.
3. To develop a sales and invoice management module that records vehicle sales, updates vehicle availability, and generates invoices.
4. To implement Role-Based Access Control (RBAC) to manage user access based on assigned roles.
5. To apply a layered Object-Oriented design using domain models, services, repositories, Flask routes, and a MySQL database.

4. Research and Development Questions

This project aims to answer the following questions:

1. How can a web-based system improve vehicle inventory management for SMEs?
2. How can the system accurately track vehicle stock movements?
3. How can the system prevent a vehicle that has already been sold from being sold again?
4. How can Role-Based Access Control improve system security and access management?
5. How can Object-Oriented Programming help organize and maintain the system?

5. Scope and Limitations

5.1 Scope

The proposed system will include the following functions:

User Authentication and Role-Based Access Control

The system will provide secure user login and logout functions. Access to system features will be controlled using Role-Based Access Control (RBAC). The system will use three main roles:

- Admin
- Manager
- Sales Staff

Users will be assigned roles through the User Roles structure. Each role will have a set of permissions that determines which system functions the user is allowed to access. Permissions will be assigned to roles through the Role Permissions structure.

Vehicle Inventory Management

Authorized users will be able to:

- Add vehicle records
- View vehicle information
- Update vehicle information
- Delete or deactivate vehicle records
- Search and filter vehicles
- Check vehicle status

Vehicle status may include:

- Available
- Reserved
- Sold
- Inactive

Stock Movement Management

The system will record important vehicle inventory movements, including:

- Stock In
- Stock Out
- Inventory Adjustment

Each movement will be linked to a vehicle and recorded with relevant details.

Customer Management

The system will allow authorized users to:

- Add customers
- View customer information
- Update customer records
- Search customer records
- Sales Management

Sales staff will be able to:

- Select an available vehicle
- Select or add a customer
- Create a sales record
- Record the sale amount
- Complete the sale

When a sale is completed, the system will update the vehicle status and create a stock-out record.

Invoice Management

The system will:

- Generate invoice records
- Assign invoice numbers
- Display invoice details
- Provide a print-friendly invoice page
- Reports

The system will provide:

- Available vehicle reports
- Sold vehicle reports
- Sales summaries
- Stock movement history

5.2 Limitations

To keep the project manageable within one term, the first version of the system will not include:

- Online payment gateways
- GPS vehicle tracking
- Mobile applications
- Multi-branch synchronization
- AI-based vehicle price prediction
- Full accounting and tax management
- Integration with external government or vehicle registration systems

6. Target Users and Stakeholders

- **Admin:** Manages users, roles, and overall system settings.
- **Manager**: Monitors vehicle inventory, sales, and reports.
- **Sales Staff:** Manages customers, sales transactions, and invoices.
- **SME Owner:** Uses reports and system information for business decisions

- **Customer:** Receives accurate vehicle and invoice information.

Note: Based on lecturer feedback, an Assistant role is being considered for the next version of the system. The Assistant would support the Owner by handling most routine operational tasks, such as monitoring inventory, customers, sales, invoices, and reports, while important final business decisions would remain with the Owner. The detailed permissions and system access for this role will be finalized during the next design revision.

7. Proposed System Features

The proposed system will contain the following main features:

1. **User Authentication** – Provides secure login and logout for system users.
2. **User Management** – Allows authorized users to create, update, activate, and deactivate user accounts.
3. **Role-Based Access Control (RBAC)** – Assigns roles to users and permissions to roles to control access to system functions.
4. **Vehicle Inventory Management** – Manages vehicle information, status, pricing, and availability.
5. **Stock Movement Tracking** – Records Stock In, Stock Out, and Inventory Adjustment activities.
6. **Customer Management** – Manages customer information used during vehicle sales.
7. **Sales Management** – Records vehicle sales and validates vehicle availability before completing a transaction.
8. **Invoice Management** – Generates and stores an invoice for each completed sale.
9. **Search and Filtering** – Allows users to search and filter vehicle and customer information.
10. **Inventory and Sales Reporting** – Provides reports about available vehicles, sold vehicles, sales, and stock movement history.

The system will also apply important business rules. Only vehicles with an **Available** status can be sold. When a sale is completed, the vehicle status will change to **Sold**, a **Stock Out** movement will be recorded, and an invoice will be generated.

8. OOP Design Plan

The proposed system will apply Object-Oriented Programming (OOP) principles to organize the system into clear and manageable components. Each class will have a specific responsibility, and the system will separate business logic, database operations, and user interface functions.

The main domain classes of the system are described below.

8.1 Vehicle

The **Vehicle** class represents a vehicle in the inventory. It will store important information such as the vehicle code, brand, model, year, color, purchase price, selling price, and current status.

The class will also support operations related to vehicle validation and status management. A vehicle may have statuses such as Available, Reserved, Sold, or Inactive.

8.2 StockMovement

The **StockMovement** class represents changes in vehicle inventory. It will record movements such as Stock In, Stock Out, and Inventory Adjustment.

Each stock movement will be connected to a specific vehicle and will contain information such as the movement type, date, reason, and user who created the record.

8.3 Sale

The **Sale** class represents a vehicle sales transaction. It will connect the vehicle, customer, sales staff, sale date, and sale amount.

The sales process will include business rules to ensure that only available vehicles can be sold.

8.4 Invoice

The **Invoice** class represents the invoice generated from a completed sale. It will contain an invoice number, sale information, issue date, and total amount.

Each completed sale will generate one invoice.

8.5 Customer

The Customer class represents a customer who purchases a vehicle. It will store customer information needed for sales and invoice records.

8.6 User

The **User** class represents a person who is authorized to access the system. It stores account information such as username, password hash, full name, email, account status, and other relevant user information.

8.7 Role

The **Role** class represents a user's level of responsibility and access within the system. The main roles are **Admin, Manager, and Sales Staff**.

8.8 Permission

The **Permission** class represents a specific action that can be performed within the system. Permissions may represent actions such as creating vehicle records, updating vehicle information, creating sales, managing users, or viewing reports.

Permissions are assigned to roles through the **Role Permissions** structure. Users receive access to system functions based on the permissions associated with their assigned roles.

8.9 UserRole

The UserRole structure represents the relationship between users and roles. It connects a user account to one or more assigned roles.

Separating user-role assignments from the User class improves flexibility and allows role assignments to be managed independently from user account information.

8.10 RolePermission

The **RolePermission** structure represents the relationship between roles and permissions. It determines which system actions are available to each role.

A role may contain multiple permissions, while the same permission may be assigned to multiple roles. This structure supports a flexible many-to-many relationship between roles and permissions.

9. System Architecture

The proposed system will use a layered architecture to separate different responsibilities. This approach will make the system easier to organize, maintain, test, and understand.

The main system flow will be:

Browser → Flask Routes → Services → Repositories → MySQL Database

9.1 Presentation Layer

The presentation layer includes the web interface that users interact with through a browser. It will use HTML templates, CSS, and JavaScript to display information and collect user input.

9.2 Route Layer

Flask routes will receive requests from users and return the appropriate web pages or responses. Routes will not contain major business rules or database queries. Instead, they will call the appropriate service classes.

9.3 Service Layer

The service layer will contain the main business rules and workflows of the system.

The main services will include:

- **AuthService:** Handles authentication and access validation.
- **UserService:** Handles user, role, and permission management.
- **VehicleService:** Handles vehicle information and status rules.
- **InventoryService:** Handles stock movements and inventory operations.

- **CustomerService:** Handles customer information and validation.
- **SalesService:** Handles vehicle sales and sales business rules.
- **InvoiceService:** Handles invoice generation and management.
- **ReportService:** Handles inventory and sales reporting.

9.4 Repository Layer

The repository layer will manage communication with the MySQL database. Repository classes will perform database operations such as creating, retrieving, updating, and deleting records.

The main repositories will include:

- **UserRepository**
- **RoleRepository**
- **PermissionRepository**
- **VehicleRepository**
- **StockMovementRepository**
- **CustomerRepository**
- **SaleRepository**
- **InvoiceRepository**

9.5 Database Layer

MySQL will be used to store and manage the system data. The database will contain tables for **users, roles, user roles, permissions, role permissions, vehicles, stock movements, customers, sales, and invoices**.

Primary keys and foreign keys will be used to maintain relationships between tables and enforce referential integrity. Unique constraints and validation rules will also be applied where necessary to maintain accurate and consistent data.

10. Database Plan

The proposed system will use **MySQL** as its database management system. The database is designed to support user authentication, Role-Based Access Control, vehicle inventory

management, stock movement tracking, customer management, sales processing, and invoice generation.

The database consists of related tables connected through primary keys and foreign keys. Junction tables are used to manage many-to-many relationships between users and roles and between roles and permissions.

The database design also applies appropriate data types, unique constraints, validation rules, and referential integrity to improve data consistency and reliability.

The final database structure is represented through an **Entity Relationship Diagram (ERD)**.

10.1 Main Tables

The proposed database contains the following ten main tables:

1. Users

The **users** table stores system user accounts. It contains information such as username, password hash, full name, email, account status, and timestamps. User passwords will be stored as secure password hashes rather than plain text.

2. Roles

The **roles** table stores the system roles used for access control. The main roles are Admin, Manager, and Sales Staff. Each role represents a different level of responsibility and system access.

3. User Roles

The **user_roles** table connects users and roles. It acts as a junction table between the users and roles tables. Its composite primary key consists of user_id and role_id.

This design allows role assignments to be managed independently from user account information.

4. Permissions

The **permissions** table stores individual actions that may be performed within the system. A permission contains information such as the permission name, system module, action, and description.

Examples may include creating vehicle records, updating vehicles, creating sales, managing users, and viewing reports.

5. Role Permissions

The **role_permissions** table connects roles and permissions. It acts as a junction table that determines which permissions belong to each role.

Its composite primary key consists of role_id and permission_id. This allows one role to contain multiple permissions and one permission to be assigned to multiple roles.

6. Vehicles

The **vehicles** table stores information about each individual vehicle in inventory. Information includes vehicle code, Vehicle Identification Number (VIN), plate number, brand, model, year, color, purchase price, selling price, and current status.

Vehicle status can be:

- Available
- Reserved
- Sold
- Inactive

The VIN, vehicle code, and plate number can be stored as unique values to help prevent duplicate vehicle records.

7. Stock Movements

The **stock_movements** table records changes in vehicle inventory. Movement types include:

- Stock In
- Stock Out

- Inventory Adjustment

Each stock movement is associated with a specific vehicle and the system user who recorded the movement. It also stores the movement date, reason, and quantity.

8. Customers

The **customers** table stores customer information used during the sales process. Information includes customer code, full name, phone number, email, address, account status, and timestamps.

One customer may have multiple sales transactions over time.

9. Sales

The **sales** table stores vehicle sales transactions. Each sale connects a vehicle, customer, and system user responsible for processing the transaction.

The table stores information such as sale code, sale date, sale price, discount amount, total amount, and sale status.

Each physical vehicle can be associated with at most one sale record. This supports the business rule that a vehicle that has already been sold cannot be sold again.

10. Invoices

The **invoices** table stores invoice information generated from sales. It includes invoice number, related sale, issue date, total amount, and creation date.

Each sale can generate only one invoice. Therefore, the sale_id relationship is unique to prevent multiple invoices from being generated for the same sale.

10.2 Main Relationships

The database contains the following main relationships:

1. **Users and Roles: Many-to-Many**
   A user can be associated with one or more roles, and a role can be assigned to multiple users. This relationship is implemented through the **user_roles** junction table.

2. **Roles and Permissions: Many-to-Many**
   A role can contain multiple permissions, and a permission can be assigned to multiple roles. This relationship is implemented through the **role_permissions** junction table.

3. **Users and Sales: One-to-Many**
   One user can process multiple sales transactions, while each sale is processed by one user.

4. **Users and Stock Movements: One-to-Many**
   One user can record multiple stock movements, while each stock movement is recorded by one user.

5. **Vehicles and Stock Movements: One-to-Many**
   One vehicle can have multiple stock movement records throughout its inventory history, while each stock movement belongs to one vehicle.

6. **Customers and Sales: One-to-Many**
   One customer can have multiple sales transactions, while each sale belongs to one customer.

7. **Vehicles and Sales: One-to-Zero-or-One**
   A vehicle may not yet have a sale record, but once sold, it can be associated with only one sale. This prevents the same physical vehicle from being sold more than once.

8. **Sales and Invoices: One-to-Zero-or-One**
   A sale may initially exist without an invoice while the transaction is being processed. Once completed, the sale generates one invoice. A unique constraint on **sale_id** prevents multiple invoices from being associated with the same sale.

10.3 Database Constraints and Integrity

The database will apply constraints to maintain accurate and consistent information.

Primary Keys:
Each main table uses a primary key to uniquely identify each record. Junction tables use composite primary keys to prevent duplicate relationships.

Foreign Keys:
Foreign keys are used to connect related tables and maintain referential integrity.

Unique Constraints:
Unique constraints will be applied to important identifiers such as usernames, vehicle codes, VINs, plate numbers, sale codes, and invoice numbers.

The **vehicle_id** in the sales table will also be unique to prevent the same physical vehicle from being associated with multiple sales. Similarly, **sale_id** in the invoices table will be unique to prevent multiple invoices from being generated for the same sale.

Data Validation:
Vehicle prices, sale amounts, and other numeric values will be validated to prevent invalid negative values.

Vehicle Status:
Vehicle status will be restricted to the defined values:

- Available
- Reserved
- Sold
- Inactive

Stock Movement Type:
Stock movement types will be restricted to:

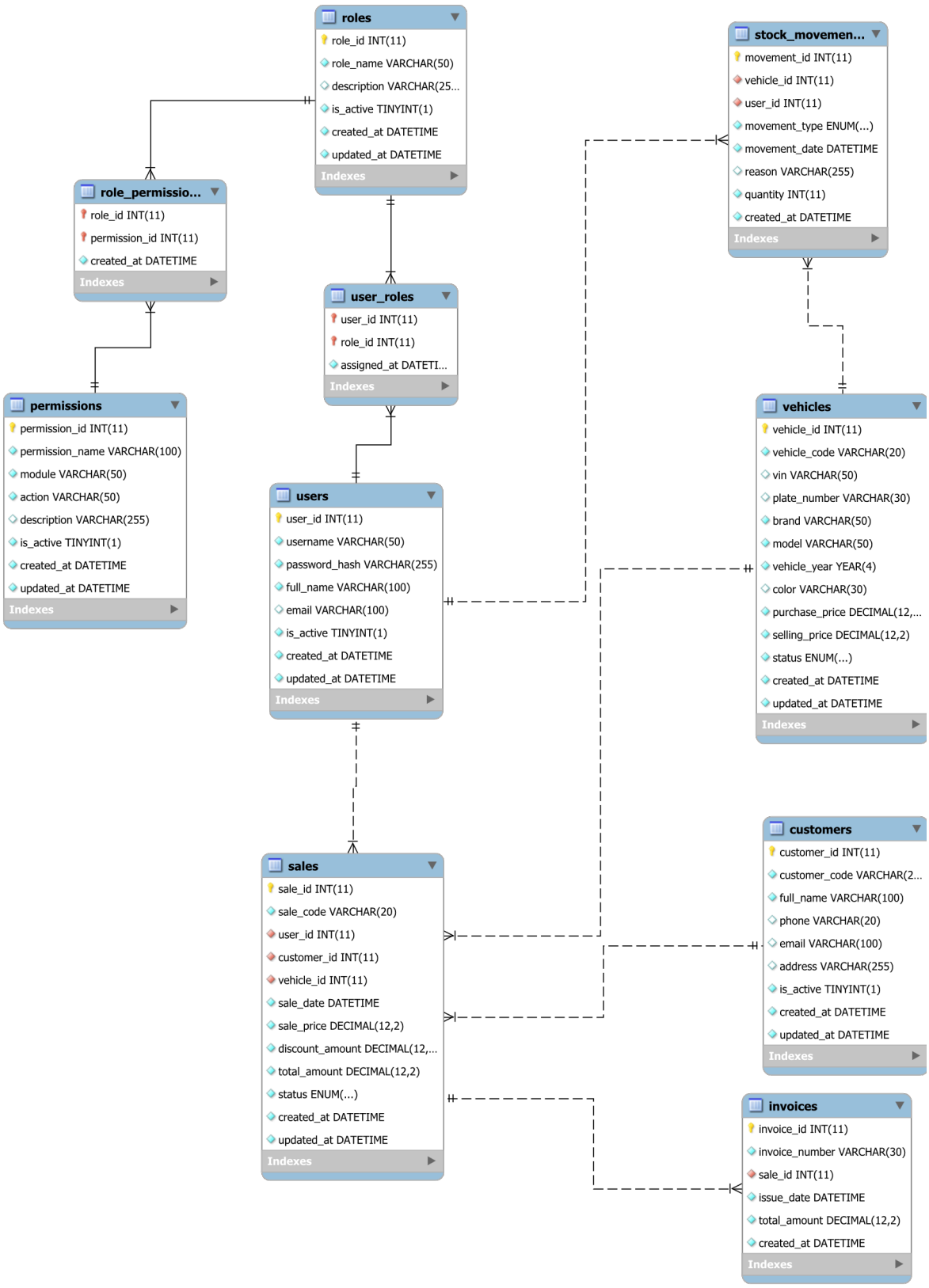
- Stock In
- Stock Out
- Adjustment

These constraints help maintain data integrity and support the business rules of the system.

10.4 Entity Relationship Diagram

The Entity Relationship Diagram (ERD) represents the database structure of the Vehicle Inventory and Sales Management System. It shows the main entities, their attributes, primary keys, foreign keys, junction tables, and relationships.

The ERD demonstrates how user access control, vehicle inventory, stock movements, customers, sales, and invoices are connected within the database.


roles
role_id INT(11)
role_name VARCHAR(50)
description VARCHAR(25...
is_active TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
stock_movemen...
movement_id INT(11)
vehicle_id INT(11)
user_id INT(11)
movement_type ENUM(...)
movement_date DATETIME
reason VARCHAR(255)
quantity INT(11)
created_at DATETIME
Indexes
role_permissio...
role_id INT(11)
permission_id INT(11)
created_at DATETIME
Indexes
user_roles
user_id INT(11)
role_id INT(11)
assigned_at DATETI...
Indexes
permissions
permission_id INT(11)
permission_name VARCHAR(100)
module VARCHAR(50)
action VARCHAR(50)
description VARCHAR(255)
is_active TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
vehicles
vehicle_id INT(11)
vehicle_code VARCHAR(20)
vin VARCHAR(50)
plate_number VARCHAR(30)
brand VARCHAR(50)
model VARCHAR(50)
vehicle_year YEAR(4)
color VARCHAR(30)
purchase_price DECIMAL(12,...
selling_price DECIMAL(12,2)
status ENUM(...)
created_at DATETIME
updated_at DATETIME
Indexes
users
user_id INT(11)
username VARCHAR(50)
password_hash VARCHAR(255)
full_name VARCHAR(100)
email VARCHAR(100)
is_active TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
customers
customer_id INT(11)
customer_code VARCHAR(2...
full_name VARCHAR(100)
phone VARCHAR(20)
email VARCHAR(100)
address VARCHAR(255)
is_active TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
sales
sale_id INT(11)
sale_code VARCHAR(20)
user_id INT(11)
customer_id INT(11)
vehicle_id INT(11)
sale_date DATETIME
sale_price DECIMAL(12,2)
discount_amount DECIMAL(12,...
total_amount DECIMAL(12,2)
status ENUM(...)
created_at DATETIME
updated_at DATETIME
Indexes
invoices
invoice_id INT(11)
invoice_number VARCHAR(30)
sale_id INT(11)
issue_date DATETIME
total_amount DECIMAL(12,2)
created_at DATETIME
Indexes

11. Methodology and Timeline

This project will follow a research and development approach. The project will begin by identifying the problem and defining the system requirements. The team will then design the system architecture, OOP structure, and database before starting development.

During development, the system will be built module by module. Testing will be performed to check whether the main features and business rules work correctly. The project will then be documented and prepared for final demonstration.

Week	Main Activity	Expected Output
1	Identify the problem and finalize project requirements	Problem definition and project scope
2	Prepare proposal and initial system design	Proposal, architecture, and initial ERD
3	Proposal presentation and feedback	Revised and approved proposal
4	Design database and OOP models	Database schema and domain classes
5	Develop authentication and RBAC	Login and role-based access
6	Develop vehicle inventory and stock movement modules	Working inventory functions
7	Develop customer and sales modules	Working sales functions
8	Develop invoice and reporting modules	Invoices and reports
9	Perform testing and fix problems	Test results and bug fixes
10	Prepare final documentation and presentation	Final report and project defense

12. Expected Results

The expected result of this project is a functional web-based Vehicle Inventory and Sales Management System for **SMEs**.

The completed system is expected to:

- Manage vehicle inventory in a centralized system.
- Track vehicle stock movements.
- Manage customer information.
- Process vehicle sales.
- Prevent sold vehicles from being sold again.
- Automatically update vehicle status after a completed sale.
- Generate invoices for completed sales.
- Control system access using roles and permissions through Role-Based Access Control (RBAC).
- Provide basic inventory and sales reports.

The project is also expected to demonstrate the use of Object-Oriented Programming through meaningful classes, clear responsibilities, service classes, and repository classes.

In addition to the working software, the project will produce testing evidence, system screenshots, database scripts, project documentation, and a final demonstration.

13. Risks and Ethics

13.1 Project Risks

Several risks may affect the development of the project.

- **Time Risk**

The team may have limited time to complete all planned features. To reduce this risk, the project will focus on the core features and follow a clear development timeline.

- **Technical Risk**

The team may face difficulties with Flask, MySQL, authentication, or implementing the OOP architecture. This risk can be reduced through research, testing, regular development, and dividing tasks among team members.

- **Data Risk**

Incorrect or invalid data may affect the accuracy of the system. The system will use validation to reduce incorrect data entry.

- **Security Risk**

Unauthorized access to the system may affect business information. Password hashing, authentication, and Role-Based Access Control will be used to improve security.

- **Team Coordination Risk**

Poor communication or unclear responsibilities may delay the project. The team will assign clear responsibilities and regularly review project progress.

13.2 Ethical Considerations

The project will consider the privacy and security of user and customer information. Passwords will not be stored as plain text, and access to sensitive information will be controlled using Role-Based Access Control (RBAC), where users are assigned roles and roles are associated with appropriate system permissions. The system will use sample or authorized data for testing and demonstration. The project will also avoid presenting generated code or content that the team cannot explain, as required by the project standard.

14. References

The final proposal will include references used during the project. These may include:

- Relevant academic and research sources.
- Teachers Instructions and class notes.

Slide Presentation

PROFESSOR: SEK SOCHEAT

Vehicle Inventory and Sales Management for (SMEs)

Advanced Object-Oriented Programming II with Python
OOP + Flask + MySQL

CSM1 Term5
MID-TERM

OUR TEAM

Lay Katsrieng

Heang Sovannara

Sour Chansokpanha

5

PROBLEM & MOTIVATION

CURRENT SITUATION

VEHICLE-SELLING SMES MAY MANAGE RECORDS USING:

- PAPER RECORDS
- SPREADSHEETS
- SEPARATE FILE

PROBLEMS

- HARD TO TRACK AVAILABLE/SOLD VEHICLES
- OUTDATED INVENTORY INFORMATION
- DIFFICULT STOCK TRACKING
- ERRORS IN SALES RECORDS
- HARD TO FIND PAST SALES/INVOICES
- LIMITED STAFF ACCESS CONTROL
- MANUAL REPORTING REQUIRED

MOTIVATION

CENTRALIZE VEHICLE INVENTORY, SALES, INVOICES, AND ACCESS CONTROL IN ONE SYSTEM.

3

OBJECTIVES

GENERAL OBJECTIVE

TO DESIGN AND DEVELOP A WEB-BASED VEHICLE INVENTORY AND SALES MANAGEMENT SYSTEM FOR SMES USING PYTHON, FLASK, MYSQL, AND OOP PRINCIPLES.

SPECIFIC OBJECTIVES

1. MANAGE VEHICLE INVENTORY USING CRUD OPERATIONS.
2. TRACK VEHICLE STOCK MOVEMENTS.
3. MANAGE CUSTOMERS, SALES, AND INVOICES.
4. CONTROL STAFF ACCESS BY ROLE.
5. APPLY A LAYERED OOP ARCHITECTURE.

4

Scope / Out-of-Scope

SCOPE

SECURITY
- Login / Logout
- Password hashing
- RBAC

INVENTORY
- Vehicle CRUD
- Search and filtering
- Vehicle status
- Stock movements

SALES
- Customer management
- Sales transactions
- Invoice generation

REPORTS
- Available vehicles
- Sold vehicles
- Sales summary
- Stock movement history

OUT-OF-SCOPE

- Online payment gateway
- GPS tracking
- Mobile application
- Multi-branch synchronization
- AI price prediction
- Full accounting system
- External government/vehicle registration integration

5

TARGET USERS & STAKEHOLDERS

ADMIN
- Manage users
- Manage vehicle records
- Manage system access
- View sales and reports

MANAGER
- Manage and monitor inventory
- Monitor stock movements
- View sales
- View reports

SALES STAFF
- View available vehicles
- Manage customers
- Create sales
- Generate invoices

SME OWNER
- Monitor business information
- Use reports for decision-making

RBAC PRINCIPLE

USER → ROLE → ALLOWED FUNCTIONS

6

PROPOSED SYSTEM FEATURES

1. Authentication & RBAC
- Login / Logout
- User management
- Role-based permissions

2. Vehicle Inventory
- Vehicle CRUD
- Search & filtering
- Vehicle status

3. Customer Management
- Customer CRUD
- Customer search

4. Sales Management
- Create sales
- Check vehicle availability
- Complete sale

5. Invoice Management
- Generate invoices
- View / Print invoices

6. Reports
- Inventory reports
- Sales reports
- Stock movement history

Business Rule
- Only available vehicles can be sold.

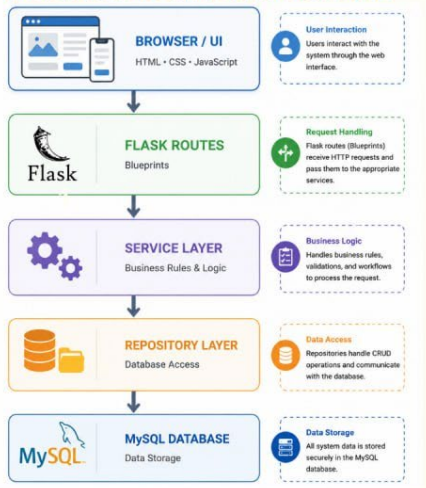
After a sale → Vehicle becomes Sold → Stock-out recorded → Invoice generated.

OOP Classes & Relationships

Class	Responsibility
Vehicle	Vehicle information and availability
StockMovement	Inventory movement history
Customer	Customer information
Sale	Vehicle sales transaction
Invoice	Sales invoice
User	System user
Role	User access level
Permission	Allowed actions

FLASK LAYERED ARCHITECTURE

PROPOSED ARCHITECTURE



RESPONSIBILITIES

ROUTES
- RECEIVE REQUESTS
- CALL SERVICES
- RETURN RESPONSES

SERVICES
- BUSINESS RULES
- VALIDATION
- WORKFLOWS

REPOSITORIES
CRUD DATABASE OPERATIONS
SQL/DATABASE COMMUNICATION

MODELS
- DOMAIN OBJECTS AND DATA

9

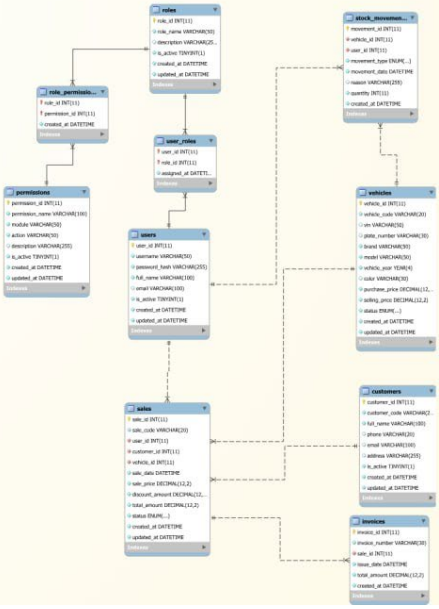
DATABASE PLAN

Main Tables

Security
- users
- roles
- permissions

Business
- vehicles
- stock_movements
- customers
- sales
- invoices

Main Relationships



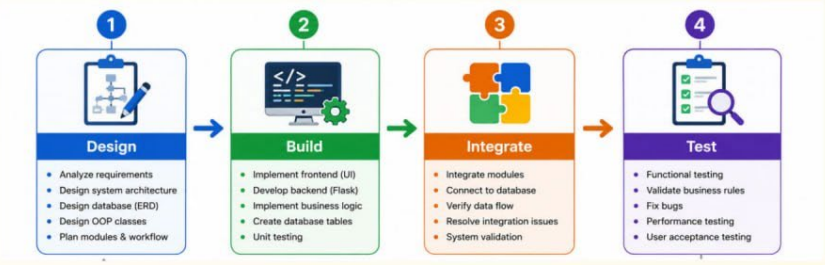
10

4-WEEK TIMELINE

Week Activities Output

Week 1: Requirements, OOP design, ERD, database setup System design & database
Week 2: Authentication, RBAC, vehicle CRUD, stock movement Inventory module
Week 3: Customer, sales, invoice Sales workflow
Week 4: Reports, testing, bug fixing, documentation Final working system

Development Flow



11

EXPECTED OUTPUTS

Working Web Application

- Secure login system
- RBAC
- Vehicle inventory management
- Stock movement tracking
- Customer management
- Sales processing
- Invoice generation
- Reports

Technical Outputs

- Flask application
- MySQL database
- OOP class structure
- Layered architecture
- ERD
- Database scripts

Expected Result:

A functional, secure, organized, and maintainable Vehicle Inventory and Sales Management System for SMEs.

12

Q&A / Lecturer Feedback

THE END

Thank You For Listening