JAWAHAR NAVODAYA VIDYALAYA
RC GHAT KHOWAI

CLASS 12 (COMPUTER SCIENCE)
SESSION: 2025–2026

PROJECT TOPIC:
TATA Lite
An Inventory Management System

Subject Teacher:
Mr. Shajen Debnath

(Signature)

Principal:
Ms. Arpita Shukla

(Signature)

GROUP 6 MEMBERS

Dhrubajyoti Chowdhury · Tanbir Chanda · Priyajit Sen
Tanmoy Adhikari · Gouri Debbarma

Acknowledgement

We take this opportunity to express our sincere gratitude to all those who have contributed, directly or indirectly, to the successful completion of our Computer Science project for the academic session **2025–2026**.

First and foremost, we would like to express our heartfelt thanks to our subject teacher, **Mr. Shajen Debnath**, for his invaluable guidance, constant encouragement, and constructive feedback throughout the development of this project. His patient supervision, clear explanations, and continuous support helped us overcome difficulties and gain a deeper understanding of the subject.

We are equally grateful to our respected **Principal, Ms. Arpita Shukla**, for providing a positive academic environment and the necessary facilities that enabled us to carry out this project successfully. Her encouragement and support have been a constant source of motivation for us.

We also acknowledge the support provided by our school, which gave us the opportunity to apply our theoretical knowledge into practical work. This project has helped us enhance our programming skills, logical thinking, and understanding of real-world applications.

Finally, we would like to thank each other as group members for our cooperation, dedication, and teamwork. The collective effort, mutual respect, and shared responsibility within the group played a vital role in the successful completion of this project.

Contents

1 Problem Statement

In many small to medium-sized retail businesses, inventory management is still performed manually using paper registers or unconnected spreadsheets. This traditional approach presents several significant challenges:

- **Data Inaccuracy:** Manual entries are prone to human error, leading to mismatches between actual stock and recorded stock.
- **Time-Consuming Calculations:** Calculating bills, taxes, and total daily profits manually is slow and inefficient, especially during peak hours.
- **Lack of Real-time Tracking:** Shop owners often struggle to know which items are running low on stock until they run out completely.
- **Data Security:** Physical registers can be lost, damaged, or easily manipulated.

The Proposed Solution:
The **"TATA Lite"** Inventory Management System is designed to address these problems by automating the core processes of a shop. It provides a digital interface to track products, manage sales transactions, generate bills instantly, and store data securely using a MySQL database backend.

2 About the Project

Overview:
Here at JNV Khowai we have learnt many wonderful things. This CS project is a culmination of what we have learnt in our CS classes in 2 years of higher secondary education.

Python and MySQL Connectivity:
Our project focuses on the minute details of the working of Python with *"TATA Lite"* being an illustrative example that relies upon the MySQL database system for data storage and manipulation.

In-depth Research:
Python is a high-level programming language, and understanding its workings is crucial for debugging. Our project brings up these workings in subsequent pages.

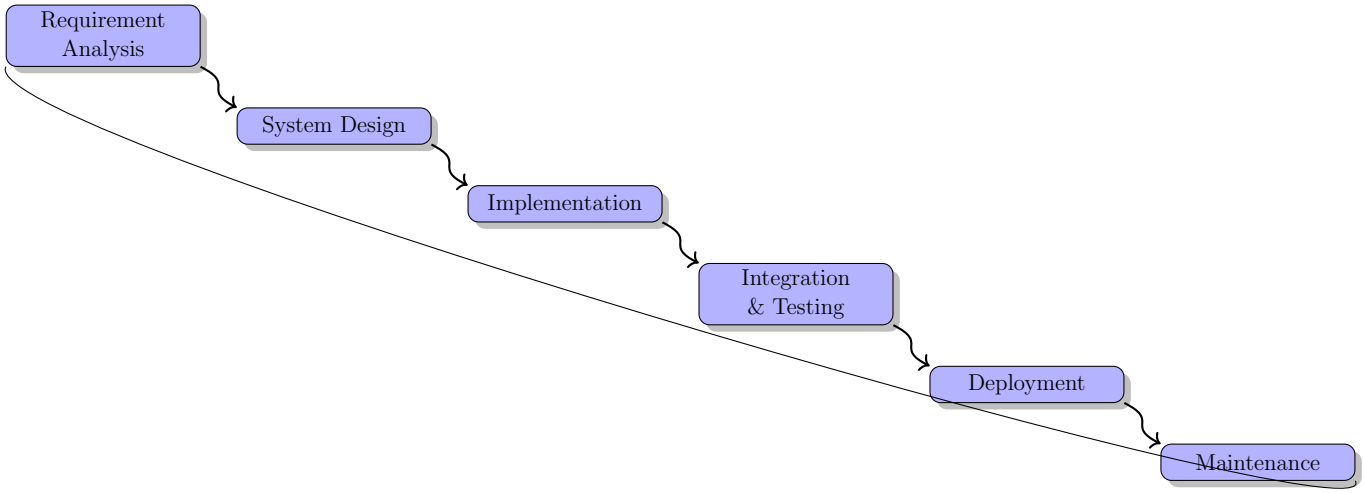
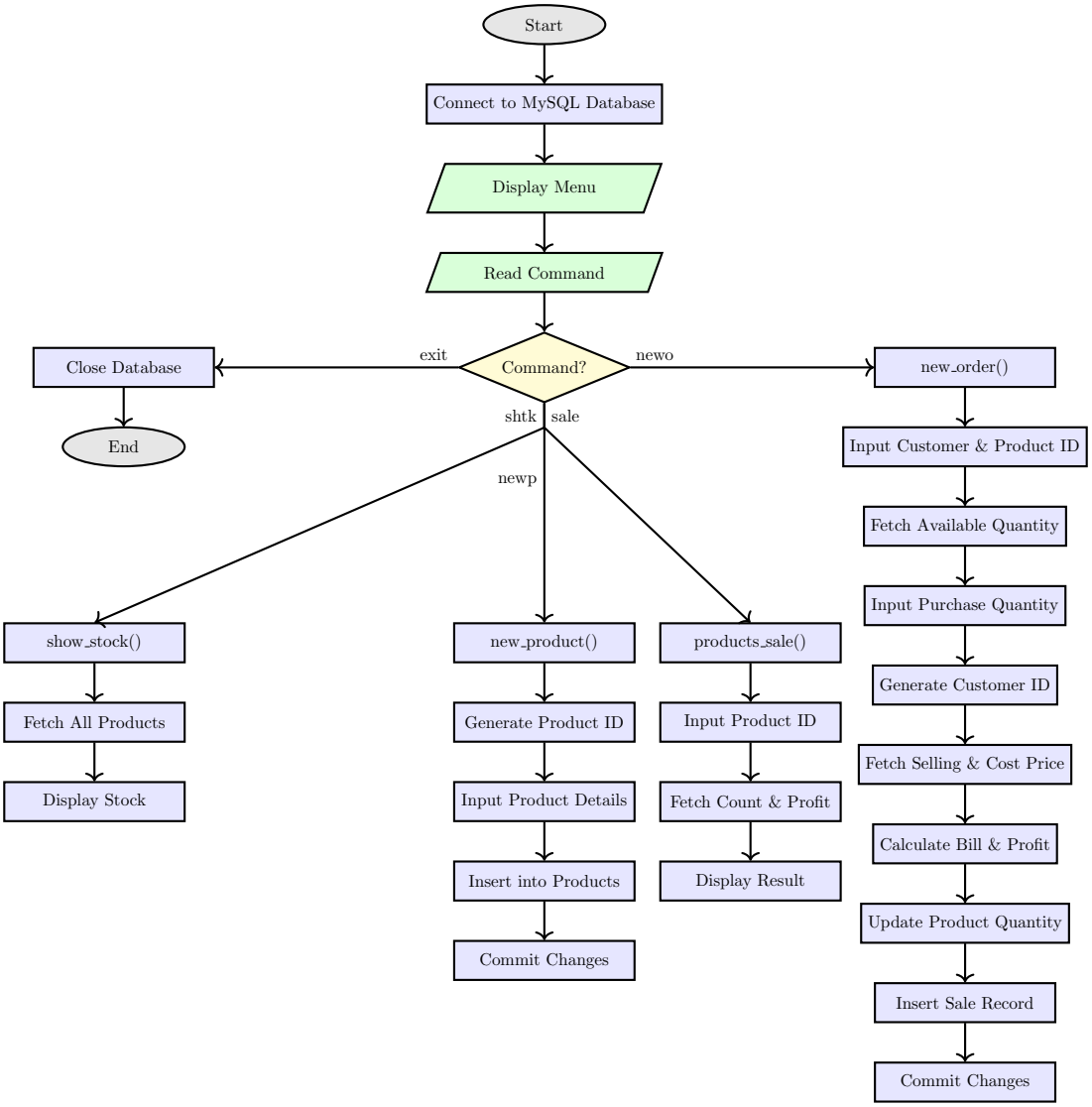


Figure 1: Waterfall model used in the project development

3 Program Flow Diagram



4 Error Analysis

During the development of the inventory management system using Python and MySQL, various errors occurred due to logical mistakes, syntax issues, incorrect data handling, and database-related problems.

Understanding these errors helped in debugging the program and improving its reliability.

Python Errors

No.	Error Name	Cause (Explanation)	Example from Program
1	NameError	Occurs when a variable or function name is used before it is defined.	Using `avail_qty` instead of `Old_Qty`.
2	TypeError	Raised when an operation is applied to incompatible data types.	`query="SELECT..." + p_Id + ';'`
3	ValueError	Occurs when a function receives an invalid value of the correct type.	`buy_Qty = int(input())` with non-numeric input
4	IndexError	Raised when accessing a list element outside its valid index range.	Accessing `queries[6]` when list size is 6
5	SyntaxError	Occurs when Python syntax rules are violated.	Missing colon in `def new_order()`
6	AttributeError	Raised when an object does not have the called method or attribute.	`cur.excute(query)`
7	ZeroDivisionError	Occurs when division by zero is attempted.	`profit = buy_Qty / 0`
8	IndentationError	Raised due to incorrect indentation of code blocks.	Improper indentation inside `while True:` loop
9	ImportError	Occurs when a required module cannot be imported.	Wrong module name in `import mysql.conector`
10	TypeError (Arguments)	Raised when function arguments are missing or extra.	Calling `products_sale()` without `p_Id`

MySQL Errors

No.	Error Name	Cause (Explanation)	Example from Program
1	OperationalError	MySQL connection fails due to wrong credentials.	Wrong password in `m.connect()`
2	ProgrammingError	Invalid SQL syntax or incorrect table/column names.	`SELEC * FROM products`
3	IntegrityError	Database integrity constraints are violated.	Duplicate `product_id` insertion
4	DataError	Invalid or out-of-range data in queries.	Inserting string into numeric column
5	InterfaceError	Database cursor or connection is misused.	Using cursor after `con.close()`
6	DatabaseError	General database-related error.	Unexpected failure during commit
7	InternalError	MySQL encounters an internal processing issue.	Cursor state error after fetch
8	NotSupportedError	Unsupported MySQL features are used.	Unsupported SQL function usage
9	DataError (Out of Range)	Numeric value exceeds the allowed range.	Inserting very large value into `DECIMAL`

5 Source Code

```
import mysql.connector as m
con = m.connect(host="localhost", user="root", passwd="root", database
    = "Group6Database")
cur = con.cursor()

def new_order():
    p_Id, c_Name, Old_Qty, buy_Qty, sp, cp = 0,0,0,0,0,0

    queries = [
        "SELECT product_quantity FROM products",
        "SELECT MAX(customer_id) FROM sales_transaction",
        "SELECT selling_price FROM products WHERE product_id = {}",
        "SELECT cost_price FROM products WHERE product_id = {}",
        "UPDATE products SET product_quantity = product_quantity - {}
           WHERE product_id = {}",
        "INSERT INTO sales_transaction VALUES ({},{},{},{},
           DATE(now()), {})"
    ]

    c_Name = input("Enter Customer Name:")
    p_Id = int(input("Enter Product ID:"))
    cur.execute(queries[0])
    data = cur.fetchone()[0]
    Old_Qty = data
    print("Available Quantity:", Old_Qty, "units.")
    cur.fetchall()
    buy_Qty = int(input("Enter Quantity to buy:"))
    cur.execute(queries[1])
    data = cur.fetchone()[0]
    last_c_Id = data
    c_Id = last_c_Id + 10
    cur.execute(queries[2].format(p_Id))
    data = cur.fetchone()[0]
    sp = data
    cur.fetchall()
    cur.execute(queries[3].format(p_Id))
    data = cur.fetchone()[0]
    cp = data
    bill = sp*buy_Qty
    cur.fetchall()
    print(
        "Your Customer ID will be:", c_Id,
        "\n Your bill:", bill,
        "\n Press any key to continue..."
    )
    input()
    profit = buy_Qty*(sp - cp)
    cur.execute(queries[4].format(buy_Qty, p_Id))
    c_Name = '"'+ c_Name +'"'
    queries[5] = queries[5].format(c_Id, c_Name, p_Id, bill, profit)

    cur.execute(queries[5])
    con.commit()
    print("You are all set!")

def new_product():
```

```
    query1 = "SELECT MAX(product_id) FROM products"
    cur.execute(query1)
    last_p_Id = cur.fetchone()[0]
    new_p_Id = last_p_Id+1
    p_Name = input("Enter product Name:")
    p_Qty = int (input("Enter product quantity:"))
    p_cp = float(input("Enter cost price:"))
    p_sp = float(input("Enter selling price:"))
    query2 = "INSERT INTO products VALUES("
    +str(new_p_Id)+','+'"'
    +str(p_Name)+'"'+','
    +str(p_Qty)+','
    +str(p_cp)+','
    +str(p_sp)+');'
    cur.execute(query2)
    con.commit()

def products_sale(p_Id):
    query="SELECT COUNT(product_id), SUM(total_profit) FROM
       sales_transaction WHERE product_Id = "+str(p_Id)+';'
    cur.execute(query)
    data = cur.fetchone()
    count=data[0]
    net_profit=data[1]
    print(count,"people bought this.Total profit:",net_profit)
    cur.fetchall()

def show_stock():
    query="SELECT * FROM products"
    cur.execute(query)
    data = cur.fetchall()
    print("ProductID    Name     Qty     Cost price     Selling
       price\n"
          "-----------------------------")
    for i in data:
        print(
            f"{i[0]:10}",
            f"{i[1]:10}",
            f"{i[2]:10}",
            f"{i[3]:10}",
            f"{i[4]:10}"
        )

menu = (
    "[exit] for exit           |MENU|"
    "\n1.---[newo]---for new_order()---place new order."
    "\n2.---[newp]---for new_product()---add new product to stock."
    "\n3.---[shtk]---for show_stock()---show everything in stock."
    "\n4.---[sale]---for products_sale(p_Id)---see total profit from a
       product product."
)

print(menu)
while True:
    cmd=input(">>>").lower()
    if cmd == "exit":
        con.close()
        break
```

```
109     elif cmd == "newo":
110         new_order()
111     elif cmd == "newp":
112         new_product()
113     elif cmd == "shtk":
114         show_stock()
115     elif cmd == "sale":
116         p_Id = input("Enter product Id to see:")
117         products_sale(p_Id)
```

6 Python Program Annotation

Table 1: User-Defined Functions Annotation

Function Name	Arguments	Return Type	Purpose and Concepts Used
`new_order()`	None	None	Handles placing a new order. Uses input/output, list of SQL queries, MySQL connectivity, arithmetic operations, string formatting, and database commit.
`new_product()`	None	None	Adds a new product to the database. Uses input(), type conversion, SQL INSERT statement, and commit().
`products_sale( p_Id)`	`p_Id` (int)	None	Displays total number of sales and net profit for a product. Uses SQL aggregation functions COUNT and SUM.
`show_stock()`	None	None	Displays all products in stock. Uses for loop and formatted output using f-strings.

Table 2: Python Features Used in the Program

Python Feature	Usage
User-defined functions	Modular programming using def
Loops	while loop (menu), for loop (display stock)
Conditional statements	if, elif, else
Lists	Used to store SQL queries
MySQL Connectivity	connect(), cursor(), execute(), fetchone(), fetchall(), commit()
Input / Output	input(), print()
Type Conversion	int(), float(), str()
String Handling	Concatenation, format(), f-strings
Arithmetic Operators	+, -, *
Comments	Single-line and multi-line comments

Table 3: SQL Queries Used in the Program

SQL Query	Purpose / Use
`SELECT product_quantity FROM products`	Fetches the available quantity of products from the `products` table.
`SELECT MAX(customer_id) FROM sales_transaction`	Finds the last customer ID to generate a new customer ID.
`SELECT selling_price FROM products WHERE product_id = {}`	Retrieves the selling price of a specific product.
`SELECT cost_price FROM products WHERE product_id = {}`	Retrieves the cost price of a specific product.
`UPDATE products SET product_quantity = product_quantity - {} WHERE product_id = {}`	Updates the stock quantity after a product is sold.
`INSERT INTO sales_transaction VALUES (...)`	Inserts a new sales record including customer ID, product ID, bill amount, date, and profit.
`SELECT MAX(product_id) FROM products`	Finds the highest product ID to generate a new product ID.
`INSERT INTO products VALUES (...)`	Adds a new product record into the `products` table.
`SELECT COUNT(product_id), SUM(total_profit) FROM sales_transaction WHERE product_Id = ...`	Calculates the number of times a product was sold and the total profit earned.
`SELECT * FROM products`	Displays all product records from the stock.

Table 4: Database Table — `products`

Field Name	Data Type	Key	Description
`product_id`	INT	Primary Key	Stores unique identification number for each product.
`product_name`	VARCHAR(20)	—	Stores the name of the product.
`product_quantity`	INT	—	Stores the available quantity of the product in stock.
`cost_price`	DECIMAL(10,2)	—	Stores the cost price of the product.
`selling_price`	DECIMAL(10,2)	—	Stores the selling price of the product.

Table 5: Database Table — `sales_transaction`

Field Name	Data Type	Key	Description
`customer_id`	INT	Primary Key	Stores unique customer ID for each sale.
`customer_name`	VARCHAR	—	Stores the name of the customer.
`product_id`	INT	Foreign Key	Stores the ID of the purchased product (linked to products table).
`bill`	DECIMAL(10,2)	—	Stores the total bill amount for the purchase.
`date_of_purchase`	DATE	—	Stores the date on which the transaction occurred.
`total_profit`	DECIMAL(10,2)	—	Stores the profit earned from the transaction.

7 Future Possibilities

This project, while currently a robust console-based application, serves as a strong foundation for more advanced software engineering concepts. As we progress to higher education and college-level studies, "TATA Lite" can be significantly extended and improved in the following ways:

- **Graphical User Interface (GUI):** Currently, the program operates on a Command Line Interface (CLI). In the future, we can integrate Python libraries like **Tkinter** or **PyQt** to create a user-friendly window-based application with buttons, input fields, and menus.
- **Web Integration (Django/Flask):** The logic used here can be adapted into a full-stack web application using frameworks like **Django** or **Flask**. This would allow the inventory to be accessed remotely from any device via a browser, moving the project from a local system to a cloud-based solution.
- **Data Visualization & Analytics:** By integrating libraries such as **Pandas** and **Matplotlib**, the system could generate visual reports, such as bar graphs of monthly sales or pie charts showing the most popular products, aiding in business decision-making.
- **Barcode & QR Code Support:** To make the "New Order" process faster and more professional, we could implement a barcode scanner interface. This would automate the entry of Product IDs, reducing human error.
- **Enhanced Security:** Future iterations could include a secure **Login/Authentication System** with hashed passwords to ensure that only authorized personnel can alter stock data or view sales history.

This project demonstrates that the core logic of database management remains consistent, whether in a simple school project or a complex enterprise system.

8 Learning Outcomes

The journey of developing "TATA Lite" was an intense and educational experience. The deadline extension to early January provided us with a small but crucial window of relief, allowing us to begin our planning in mid-December.

1. Planning and System Design

We started with a rough idea that was gradually shaped into a concrete plan. By dividing our thoughts and referencing work done by our seniors, we sketched out the entire project within a limited timeframe. Within five days, we developed our synopsis, which was promptly accepted by our subject teacher.

During this phase, we evaluated **four different approaches**, weighing them carefully based on complexity and purpose. Ultimately, we chose the path of simplicity: a final draft consisting of just **two MySQL tables and four Python functions**. All logic was user-defined and meticulously planned on paper before touching the keyboard.

2. Implementation Strategy

We adopted a disciplined approach by first writing all codes neatly on paper.

- **Database Creation:** We successfully built our database—proudly naming it `Group6Database`—a the required tables in about one hour. The process was smooth, with the only minor hurdles being the installation of the MySQL server and downloading the connector.
- **Coding Phase:** The paper-based logic was typed into the system the next day, transforming our draft into a working script.

3. Debugging and Refinement

Over the following four days, we worked continuously to debug and refine the program. As testing revealed errors, we fixed them one by one. Crucially, we slowed down to carefully note these errors, which later served as the specific references for the **Error Analysis** section of this documentation.

4. Skill Development

Each day of this project offered a new lesson. Beyond technical skills in MySQL, Python, and database connectivity, we gained invaluable experience in:

- **Teamwork:** Learning how to discuss problems, share responsibilities, and work as a cohesive unit.
- **Time Management:** Delivering a complete project within a tight schedule by managing our "imagination and thoughts" effectively.
- **Collaboration:** Each of us experienced something we did not know before and might never have known had we not worked together.

This project marked itself as an enriching and meaningful learning opportunity, providing us with experiences that will surely help us in the future.

Index

"Those who can imagine anything, can create the impossible."
— Alan Turing