ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

# BSc (Hons) Artificial Intelligence and Data Science

## Module: CM1601 programming fundamentals

## Individual Coursework Report

**Module Leader: Mr. Cassim Farook**


RGU Student ID : 2425472

IIT Student ID : 20240892

Student Name : Lidiya Rajapakse

# Table Of Contents



# Table Of Figures

# 1. INTRODUCTION

This report outlines the design and functionality of two interconnected computer systems: a Point-of-Sale (POS) System and a Government Tax System. Both systems were designed to carry out real-world tasks such as item and transaction management, bill generation, and tax information processing. The POS system is designed to process transactions of customers, generate bills, and export transaction data. On the other hand, the Government Tax System is responsible for importing tax transaction data, validating the data, calculating profits and tax amounts, and generating detailed tax reports.

Coding best practices, version control, and testing best practices were followed while developing these systems to ensure that both systems are robust and function as required. This report uses elaborate descriptions, flowcharts, class diagrams, test cases, and screenshots to represent the process of developing and testing the systems. The use of generative AI tools like ChatGPT helped to ease the process of coding, providing recommendations on complex tasks like checksum validation and file handling.

## 2.GitHub Repository URL and Screenshots

Private GitHub Repository URL:
https://github.com/LidiyaRajapakse/pos-tax-system

Screenshots

1.Project Structure

This screenshot demonstrates the entire project directory structure, emphasizing how files are organized in the repository.

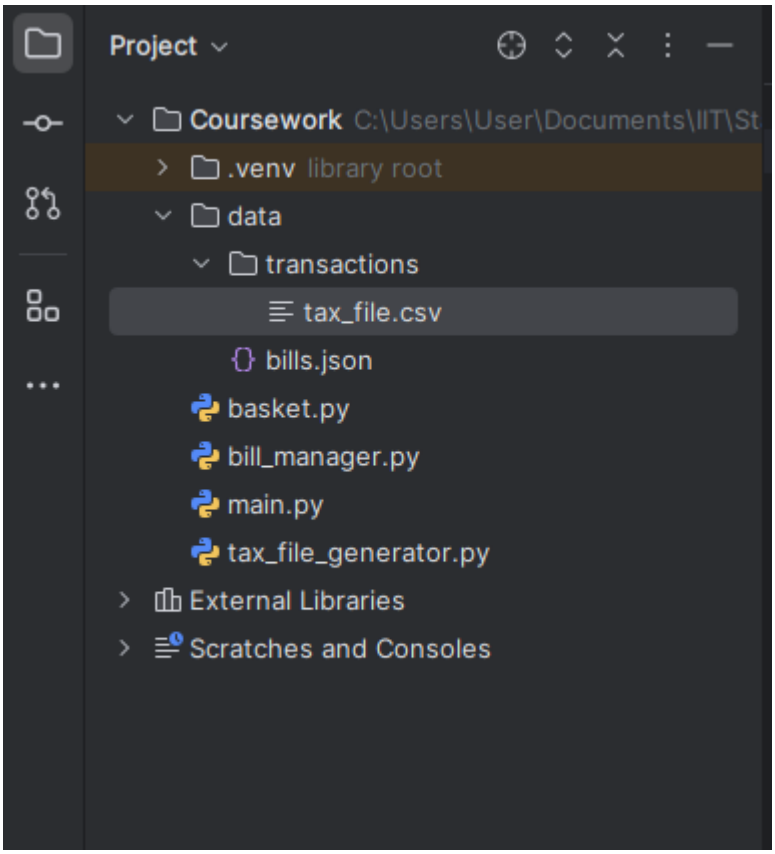


*Figure 1:Project structure of POS system*

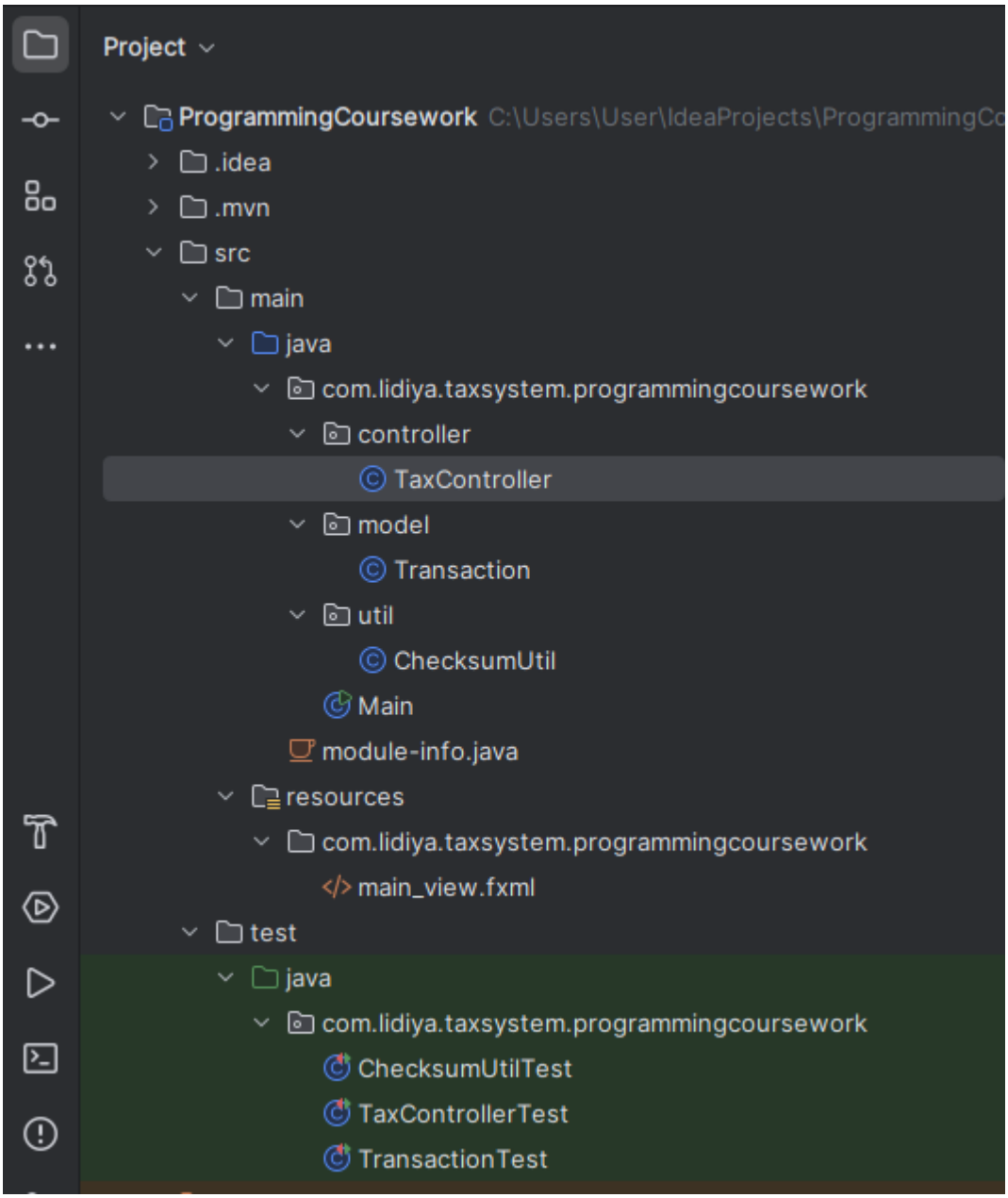


*Figure 2:Project structure of JFXGTDS*

## 2.Commit History

This screenshot will show the commit history to highlight the progress of the project over time, showcasing the key milestones and major updates.

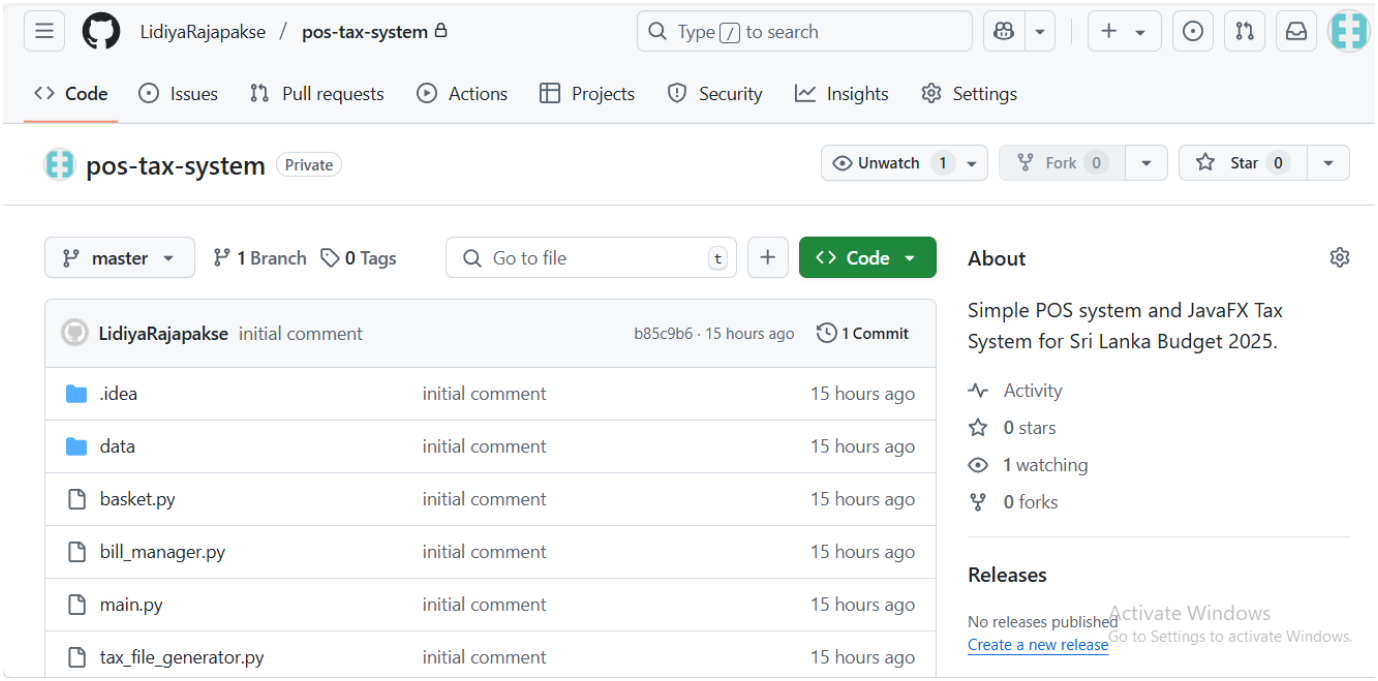


*Figure 3:Commit history*

## 3.Source code

Including a screen capture of the source code for the POS system and Government Tax system. This is to show the critical code that forms the core logic of both systems.

- POS system source code

In this section,  a screenshot  of the source code of the POS System provided to identify key sections such as:

**Item Management:** The part of the code that adds, updates, or deletes items.

```python
def add_item(self, item_code, internal_price, discount, sale_price, quantity):  Lidiya
    line_total = sale_price * quantity
    item = {
        "line_no": self.counter,
        "item_code": item_code,
        "internal_price": internal_price,
        "discount": discount,
        "sale_price": sale_price,
        "quantity": quantity,
        "line_total": line_total
    }
    self.items.append(item)
    self.counter += 1
    print(f"Item added: {item}")

def delete_item(self, line_no):  LidiyaRajapakse
    self.items = [i for i in self.items if i["line_no"] != line_no]

def update_item(self, line_no, sale_price, discount, quantity):  LidiyaRajapakse
    for item in self.items:
        if item["line_no"] == line_no:
            item["sale_price"] = sale_price
            item["discount"] = discount
            item["quantity"] = quantity
            item["line_total"] = sale_price * quantity
            break
```



*Figure 4:Item management*

**Bill Generation:** Shows the code used to generate the bill, including the methods used for calculating totals, adding discounts, and printing/exporting the final bill.

```python
def generate_csv(items, filename="data/transactions/tax_file.csv"):  1 usage  LidiyaRajapakse
    """Generate CSV file with checksum at the end."""
    os.makedirs( name: "data/transactions", exist_ok=True)
    with open(filename, "w", newline='') as f:
        writer = csv.writer(f)
        # Writing header for CSV
        writer.writerow(["item_code", "internal_price", "discount", "sale_price", "quantity", "checksum"])
        for item in items:
            # Create CSV line for each item
            line = [item["item_code"], item["internal_price"], item["discount"], item["sale_price"], item["q
            # Calculate checksum for the line
            checksum = calculate_checksum(','.join(map(str, line)))
            # Append checksum to the line and write to CSV
            writer.writerow(line + [checksum])

def generate_tax_file(items):  2 usages  LidiyaRajapakse
    """Generates a tax file in CSV format."""
    generate_csv(items)
```

*Figure 5:Bill generation*

- Government Tax System Source Code

Providing here  screenshots of the source codes of the Government Tax System. Highlighting key classes like:

TaxController: This class holds the import logic, validation, transaction processing, and tax calculation logic. Highlighting the methods that are in charge of tax calculation, file import, and data validation.

```java
public class TaxController {  LidiyaRajapakse

    @FXML private TableView<Transaction> transactionTable;
    @FXML private TableColumn<Transaction, String> itemCodeColumn;
    @FXML private TableColumn<Transaction, Double> internalPriceColumn;
    @FXML private TableColumn<Transaction, Double> discountColumn;
    @FXML private TableColumn<Transaction, Double> salePriceColumn;
    @FXML private TableColumn<Transaction, Integer> quantityColumn;
    @FXML private TableColumn<Transaction, Integer> checksumColumn;
    @FXML private TableColumn<Transaction, Double> profitColumn;
    @FXML private TableColumn<Transaction, Double> taxColumn;

    private final ObservableList<Transaction> transactions = FXCollections.observableArrayList

    @FXML  LidiyaRajapakse
    public void initialize() {
        itemCodeColumn.setCellValueFactory(new PropertyValueFactory<>( s: "itemCode"));
        internalPriceColumn.setCellValueFactory(new PropertyValueFactory<>( s: "internalPrice")
        discountColumn.setCellValueFactory(new PropertyValueFactory<>( s: "discount"));
        salePriceColumn.setCellValueFactory(new PropertyValueFactory<>( s: "salePrice"));
        quantityColumn.setCellValueFactory(new PropertyValueFactory<>( s: "quantity"));
        checksumColumn.setCellValueFactory(new PropertyValueFactory<>( s: "checksum"));
        profitColumn.setCellValueFactory(new PropertyValueFactory<>( s: "profit"));
        taxColumn.setCellValueFactory(new PropertyValueFactory<>( s: "tax"));
```

*Figure 6:TaxController.java*

ChecksumUtil: Showing how this utility is used to check the integrity of the tax transaction data.

```java
public class ChecksumUtil {  16 usages  LidiyaRajapakse
    public static int calculateChecksum(String input) {  7 usages  LidiyaRajapakse
        // Count the number of uppercase letters, lowercase letters, and numeric digits
        int caps = input.replaceAll( regex: "[^A-Z]",  replacement: "").length();
        int lows = input.replaceAll( regex: "[^a-z]",  replacement: "").length();
        int nums = input.replaceAll( regex: "[^0-9.]",  replacement: "").length();
        return caps + lows + nums;
    }

    public static boolean isValidItemCode(String code) {  7 usages  LidiyaRajapakse
        // Check if the item code contains only alphanumeric characters or underscores
        return code.matches( regex: "^[A-Za-z0-9_]+$");
    }
}
```

*Figure 7:Checksumutil*

## 3. GENERATIVE AI SOFTWARE USAGE

### AI Software Usage:

Generative AI was used in this project.

- **Software/Product Name**: *ChatGPT by OpenAI (version GPT-4)*
- **URL of the Product**: https://chat.openai.com

### Prompts used:

Prompt 1:
"How to create a simple POS system in Python with item add, update, and billing features?"

Prompt 2:
"How can I export a Python list of transactions into a CSV file?"

Prompt 3:
"Can you show me how to validate a CSV file using Java?"

Prompt 4:
"How to write a checksum function in Java for a text file?"

Prompt 5:
"How do I import a CSV file into a JavaFX table view?"

Prompt 6:
"Can you help me design a class diagram for a small tax management system?"

Prompt 7:
"What are good test cases for checking a POS system's item management feature?"

Prompt 8:
"How to calculate total profit and tax from sales data in Java?"

## 4. FLOW CHART FOR POS SYSTEM

**1.Start**
**2.Display Main Menu**
Options: Add Item / Update Item / Delete Item / Generate Bill / Export Tax File / Exit
**3.Add Item**
Enter Item Name, Code, Price, Quantity
Save item to the item list
Return to Main Menu
**4.Update Item**
Select item to update
Modify Name, Price, or Quantity
Save changes
Return to Main Menu
**5.Delete Item**
Select item to delete
Remove from item list
Return to Main Menu
**6.Generate Bill**
Select items for the bill
Apply Discounts if applicable
Show final bill amount
Save bill
Return to Main Menu
**7.Export Tax Transaction File**
Export transaction data to CSV, JSON, TSV, or XML
Generate checksum for each transaction
Save file
Return to Main Menu
**8.Exit**
**9.End**

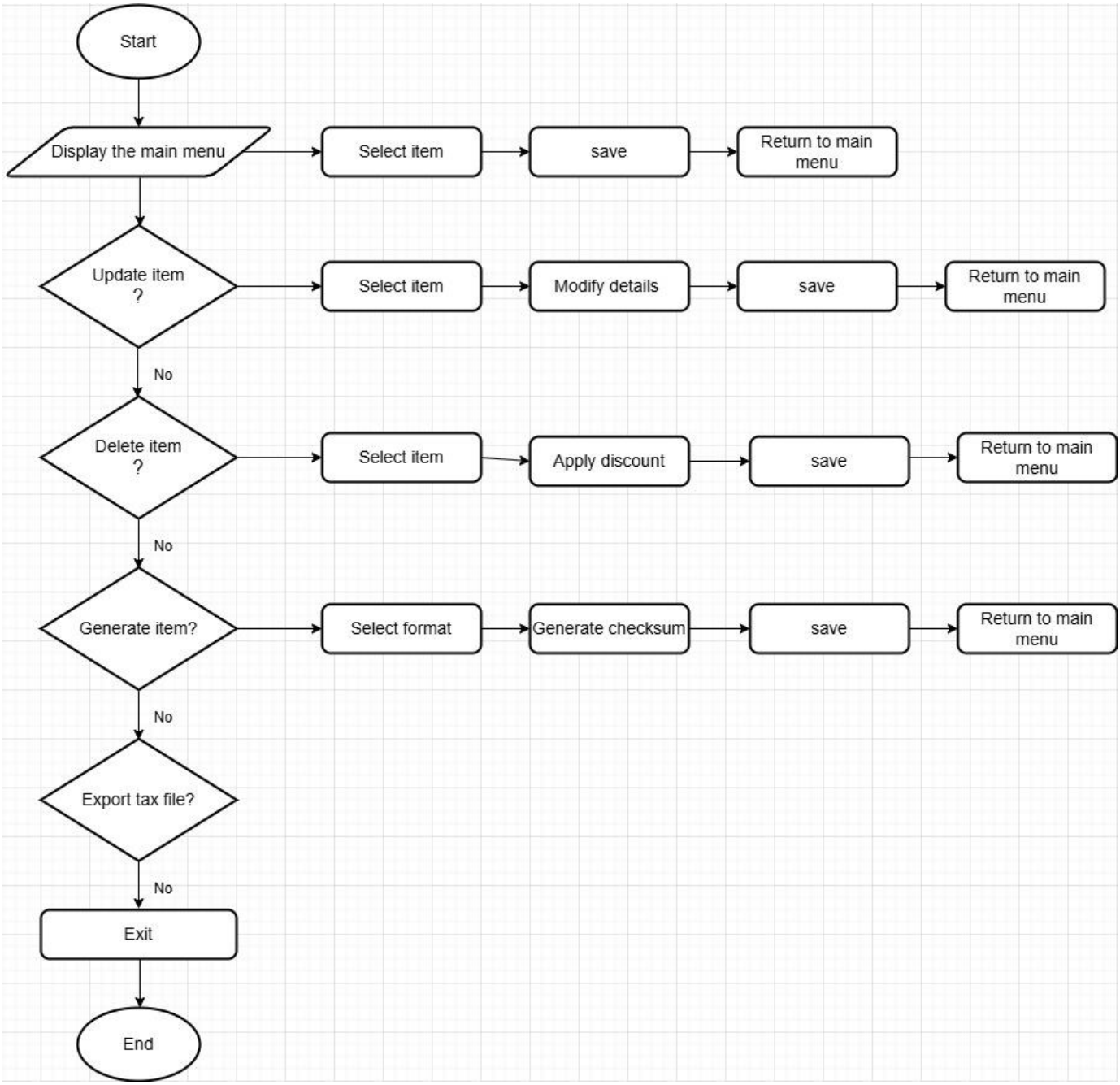


*Figure 8:Pos flow chart*

## 5.Flow Chart for Government Tax System

1.**Start**
2.**Open Application**
3.**Display Main Menu**
Options: Import Tax File / Validate Transactions / Process Profits / Calculate Taxes / Generate Tax Report / Exit
4.**Import Tax Transaction File**
Select file (CSV )

Load data into system
Return to Main Menu

5. **Validate Transactions**
   Check data format and checksum for each transaction
   Mark valid and invalid transactions
   Allow manual correction or deletion
   Return to Main Menu
6. **Process Profits**
   Calculate profit for each transaction (sale price - cost price)
   Delete transactions with zero or negative profit (if required)
   Return to Main Menu
7. **Calculate Taxes**
   Apply tax rates to calculate final tax on profits
   Return to Main Menu
8. **Generate Tax Report**
   Display summary of profits, taxes, and invalid transactions
   Allow export of final report
   Return to Main Menu
9. **Exit**
10. **End**

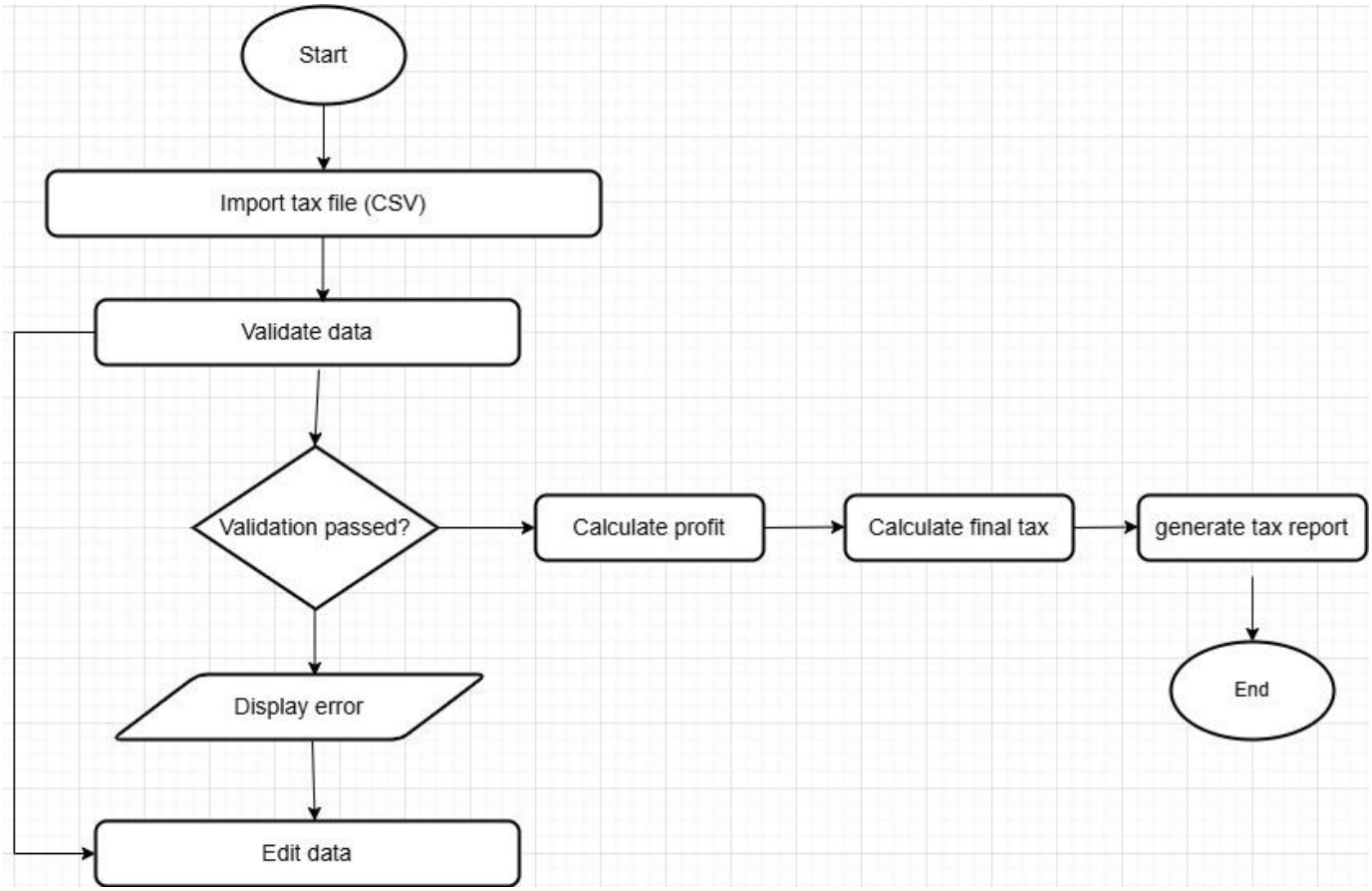


*Figure 9:Tax system flow chart*

# 6.CLASS DIAGRAM FOR GOVERNMENT TAX SYSTEM

The class diagram should illustrate the structure and relationship of the core classes in the Government Tax system.
Below is an overview in general:

Transaction: A transaction object with attributes like item code, sale price, quantity, and checksum.

TaxController: Manages the logic of processing a transaction and calculating taxes.

ChecksumUtil: Checks that each transaction has a valid checksum to prevent errors.

Main: The main entry point of the application, where the application starts.

This chart demonstrates how all the different parts of the system relate to each other.

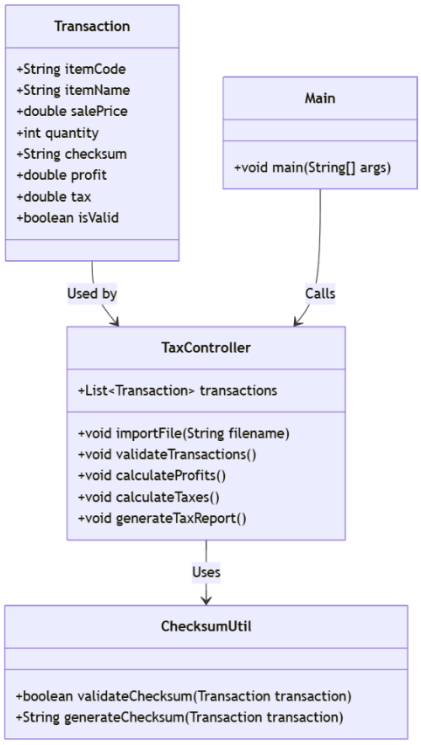


*Figure 10:Class diagram*

# 7.TEST CASE FOR POS SYSTEM

**Test Case 1**

Function to Test: Adding an item to POS System
Test Description: Test adding the item to the system when proper information is being entered.
Preconditions: POS system in run mode, and items list is not available.

Test Steps:

- Opening the POS system.
- Clicking on 'Add Item'.
- Providing correct item details
- Clicking 'Save'.

Expected Result: The item is added to the list and appears correctly.
Actual Result: Item added successfully .
Pass/Fail: Pass

```
--- POS Menu ---
1. Add Item
2. Delete Item
3. Update Item
4. View Basket
5. Generate Bill
6. Search Bill
7. Generate Tax File
8. Exit
Choose: 1
Item Code: 002
Internal Price: 600
Discount: 200
Sale Price: 100
Quantity: 12
Item added: {'line_no': 1, 'item_code': '002', 'internal_price': 600.0, 'discount': 200.0, 'sale_price': 100.0, 'quantity': 12, 'line_total': 1200.0}
Item added: 002, Price: 100.0, Quantity: 12, Total: 1200.0
```

*Figure 11:Adding an item*

# 8.TEST CASE FOR GOVERNMENT TAX SYSTEM

**Test Case 1**

Function to Test: Import and Validate Tax Transaction File
Test Description: Ensure the tax transaction file is imported and validated correctly.
Preconditions: Tax Transaction file is ready and in CSV/JSON format.

Test Steps:

- Opening the Government Tax system.
- Clicking on 'Import File'.
- Selecting the valid tax transaction file (CSV).
- Validating the file.

Expected Result: File is imported and validated successfully.
Actual Result: File imported and shows the item details and validated successfully.
Pass/Fail: Pass

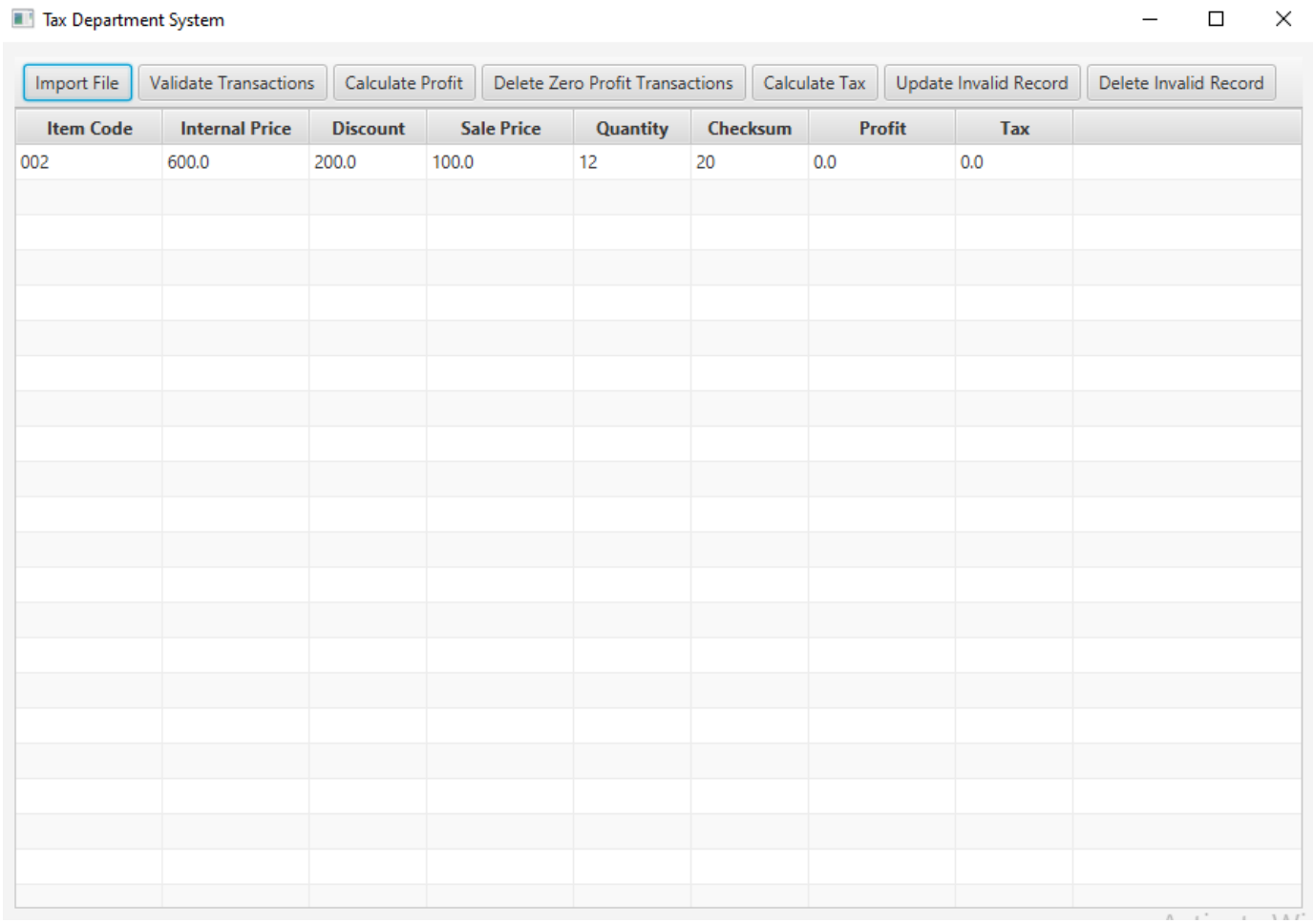


*Figure 12:Tax file imported*

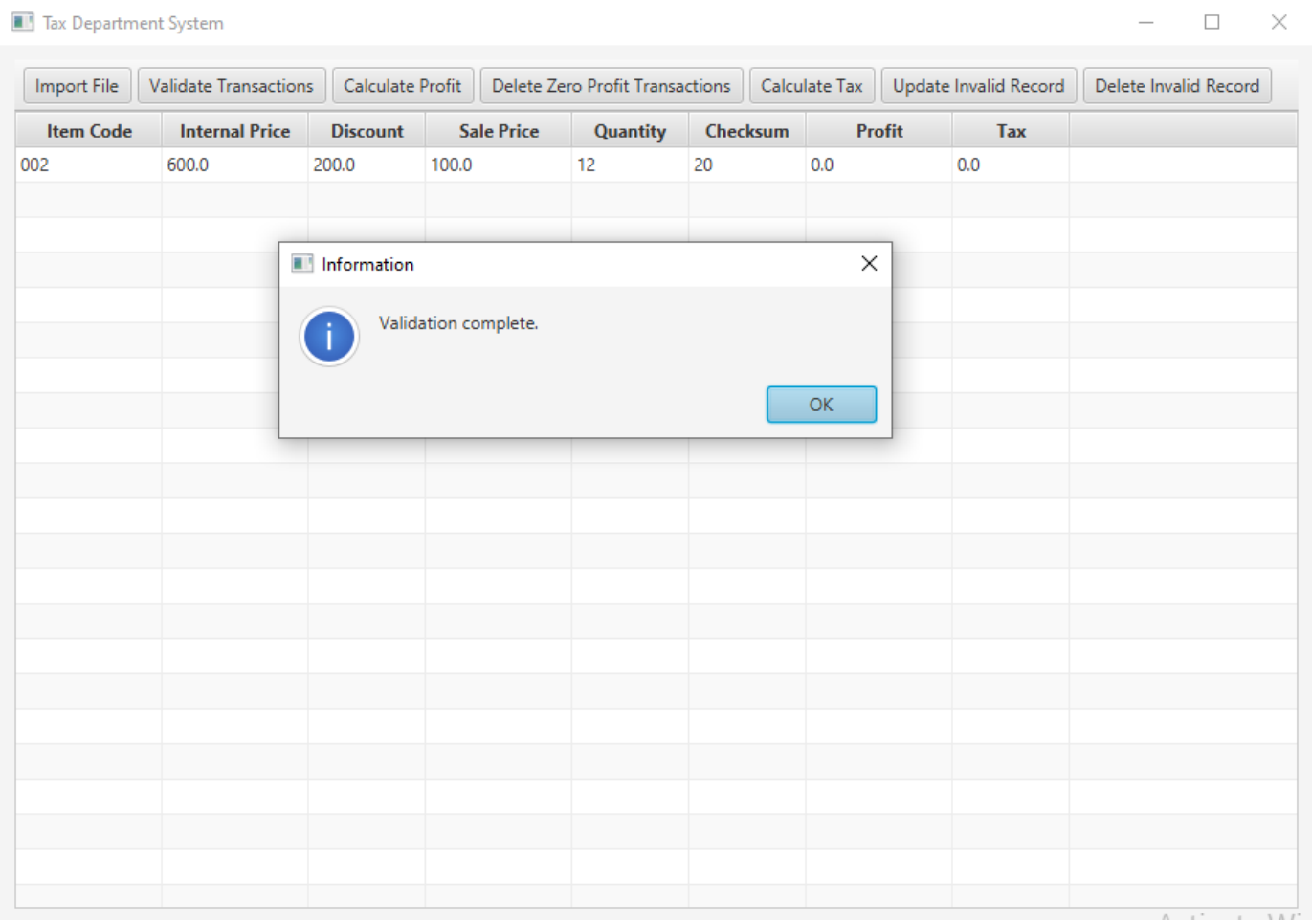


*Figure 13:Validated successfully*

## 9.THREE SCREENSHOTS OF THE POS SYSTEM IN ACTION

### Screenshot 1:

Shows the POS system interface during the addition of an item. This will show the "Add Item" form where the user will provide item information.

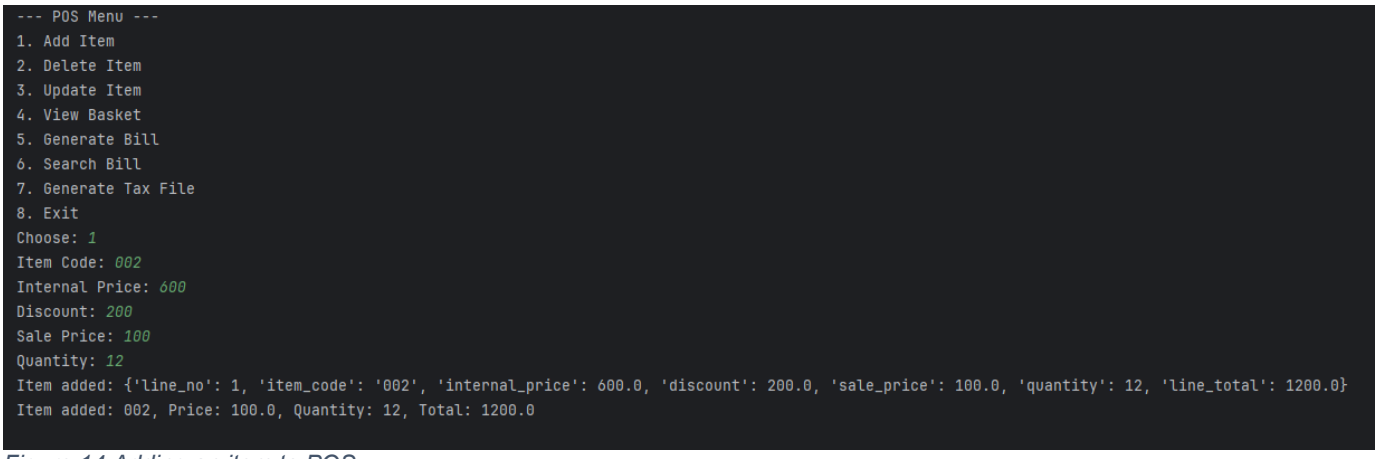


*Figure 14:Adding an item to POS*

### Screenshot 2:

Shows the POS system interface with the bill generated and items listed. This screenshot will reveal how the bill looks after the items have been added.


*Figure 15:Generating the bill*

### Screenshot 3:

Shows the POS system interface with the Tax Transaction File export feature. On this screen, the "Export" option will be shown for preparing the transaction file.

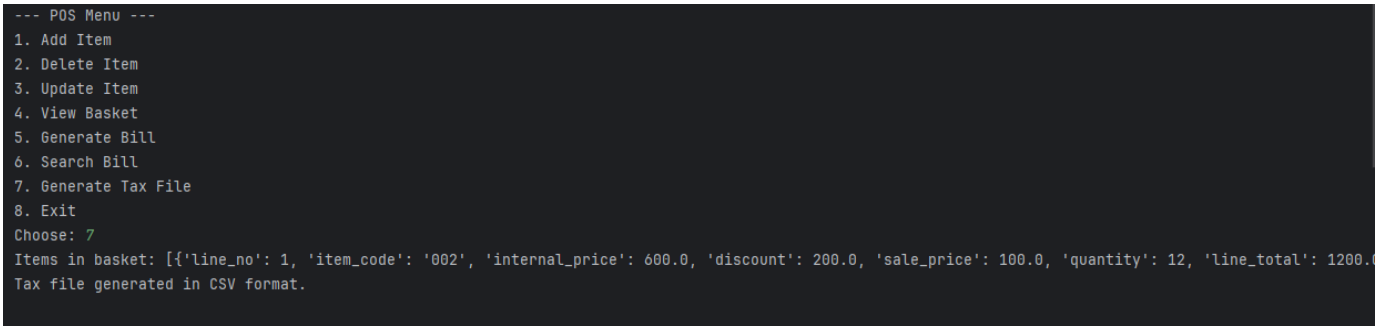


*Figure 16:Exporting the tax file*

# 10. Three Screenshots of the Government Tax System in Action

## Screenshot 1:

Show the Government Tax System interface when importing the Tax Transaction file.

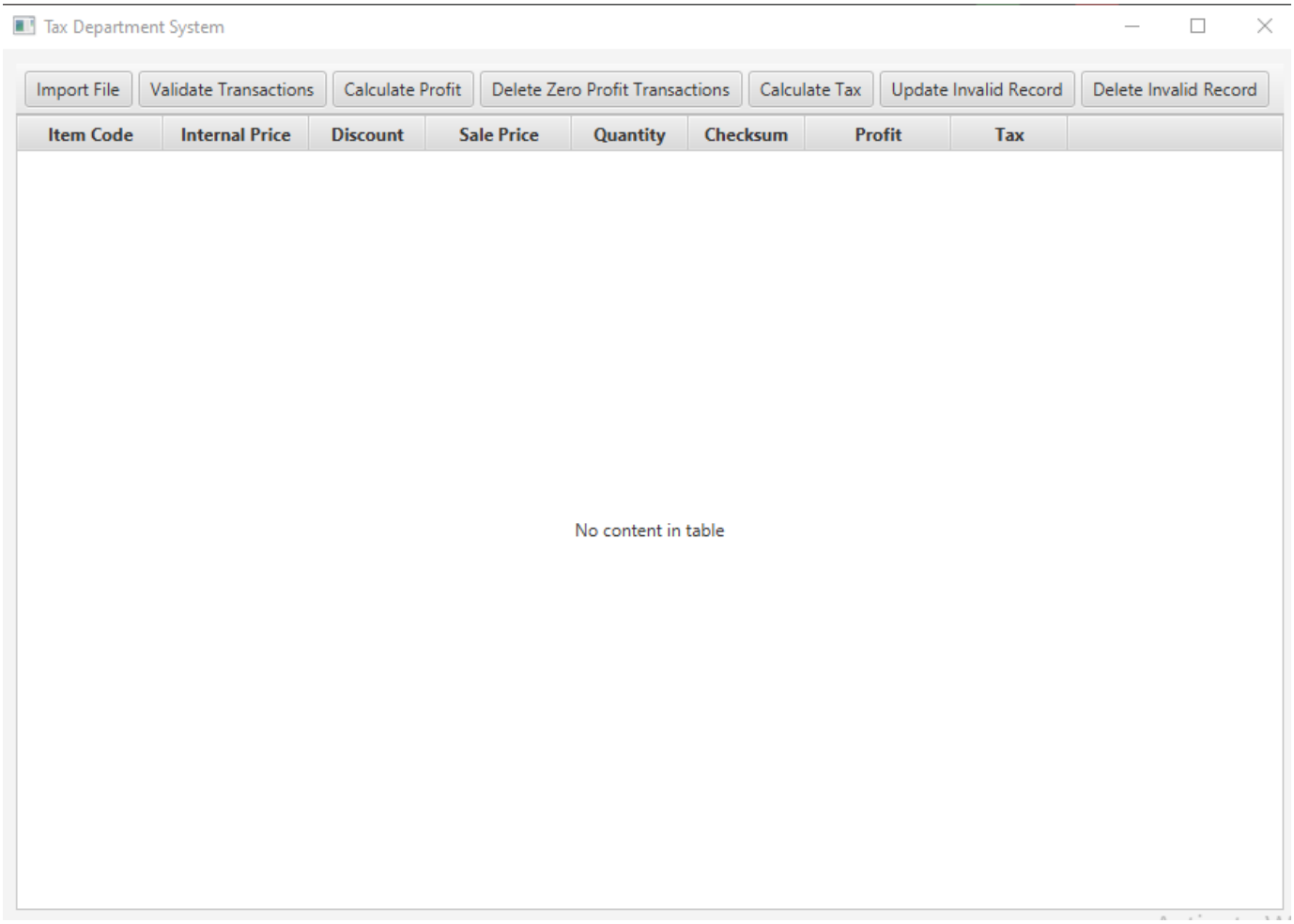


*Figure 17:Tax system*

## Screenshot 2:

Show the validation result of the imported Tax Transaction file

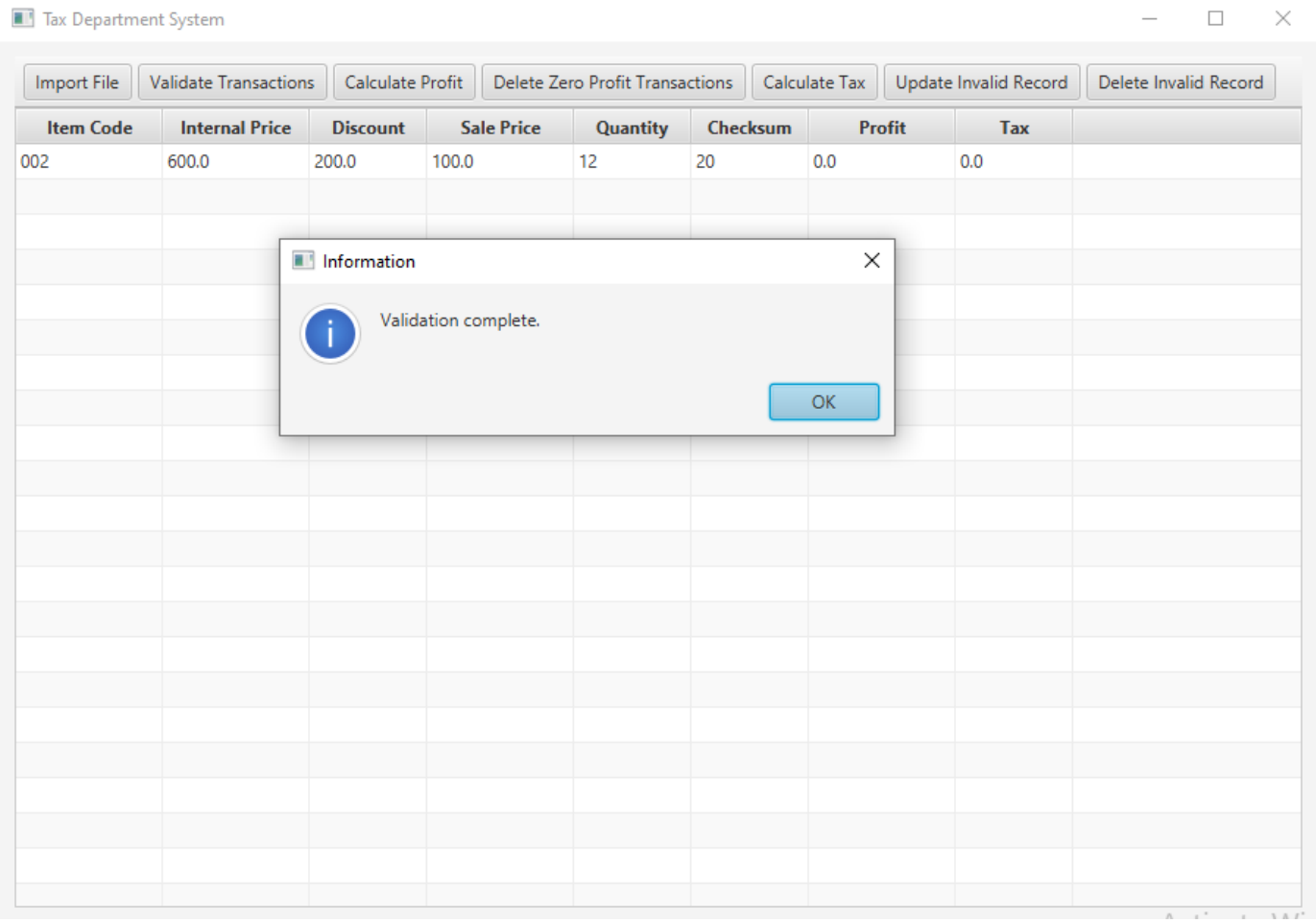


*Figure 18:Validation*

## Screenshot 3:

Show the computed tax and profits on the final report.

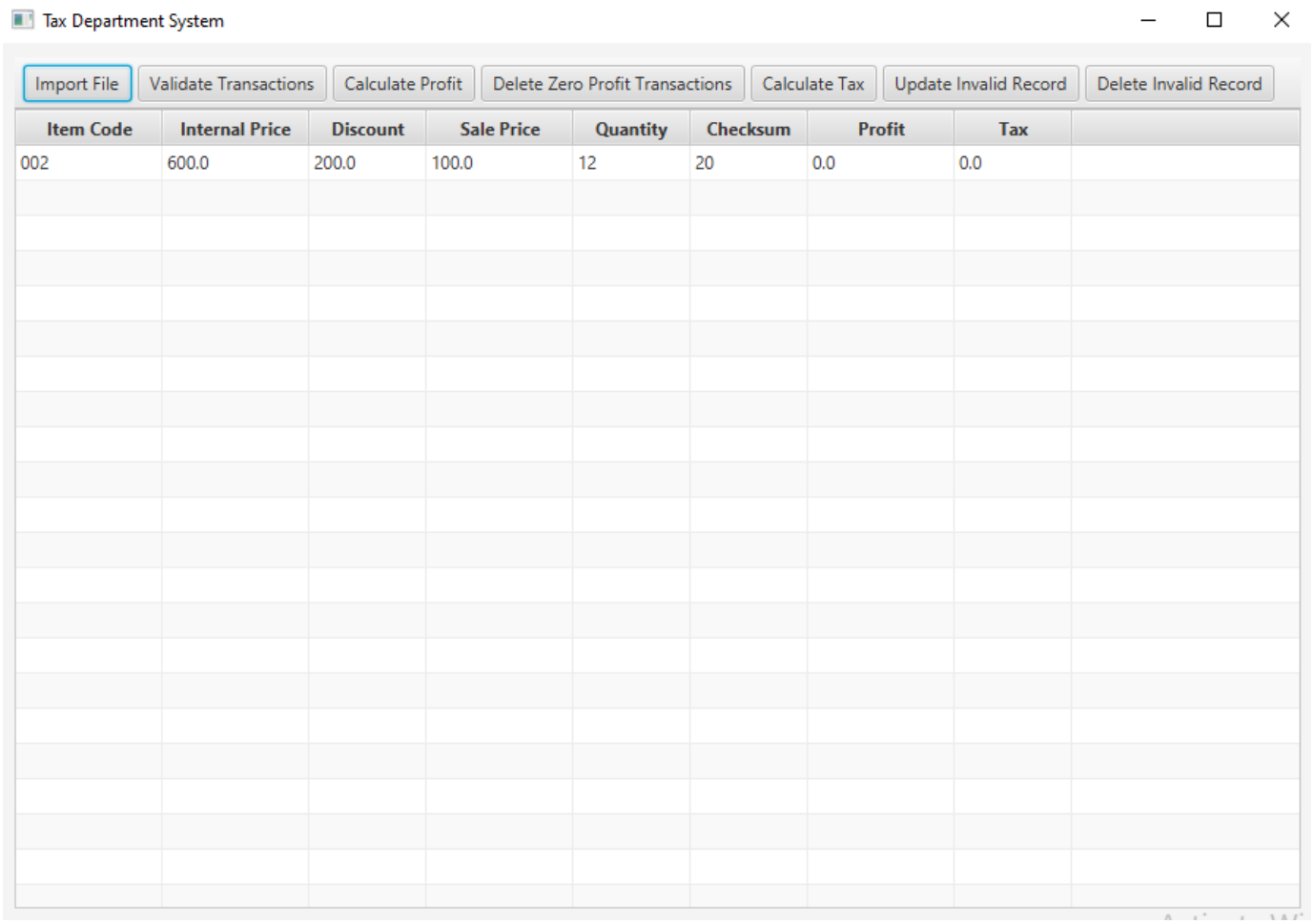


*Figure 19:Tax and profit*

## 11. Screenshot of the Data in the Tax Transaction File

Show a screenshot of the tax transaction file (CSV) imported to java tax file calculating application. Each record contains values for Item code,Internal price, Discount , Sale price , Quantity , Checksum , Profit and Tax .

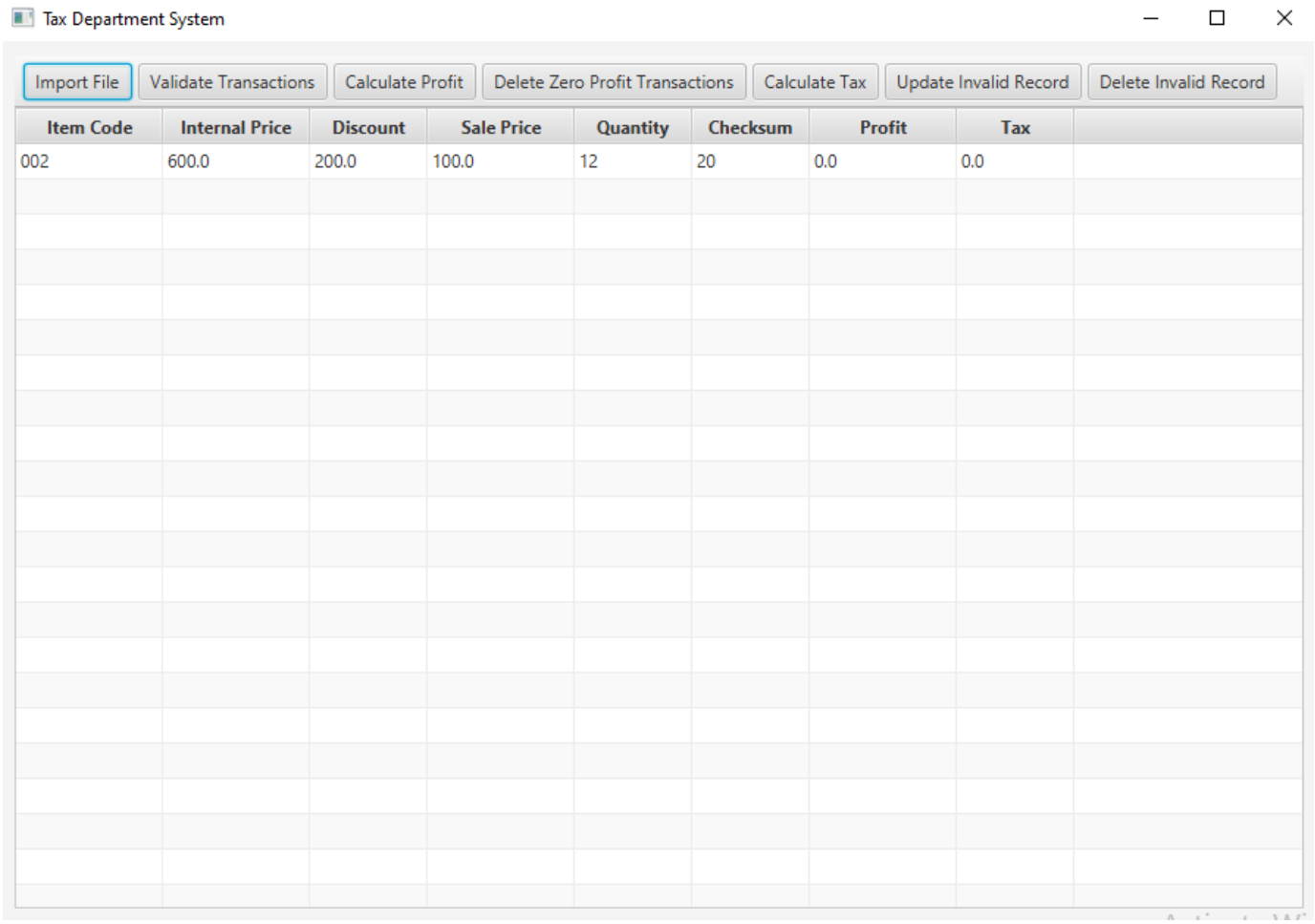


*Figure 20:Tax file calculator*

## 12. CONCLUSION

In short, the POS System and Government Tax System development show an end-to-end solution to actual problems related to sales transactions and tax administration. Through integrating item management, bill generation, and transaction validation, both systems show efficient solutions to their respective tasks. The project demonstrates the ability to manage various aspects of software development, from conceptualization and coding to testing and deployment.

This paper not only commemorates the technical victory but also highlights the importance of adequate documentation, rigorous testing, and utilization of AI tools to increase productivity. In the future, additional optimizations and new features could be incorporated into both systems to enhance user experience, increase performance, and adapt to evolving business requirements.

# 13. REFERENCES


- GitHub, Managing remote repositories. [online] Available at: https://docs.github.com/en/repositories [Accessed 27 April 2025].
- OpenAI, 2025. ChatGPT by OpenAI (GPT-4). [online] Available at: https://chat.openai.com [Accessed 27 April 2025].
- Oracle, Introduction to JavaFX for Java programmers. [online] Available at: https://openjfx.io/ [Accessed 27 April 2025].
- Python Software Foundation, 2024. Python 3 Documentation. [online] Available at: https://docs.python.org/3/ [Accessed 27 April 2025].