Mini Project Report

Project Title: **Invoice Management System**

Team Members: ***GANESH KUMAR YADAV (Name & Roll No) 27600123122***

Md AIYAZ AHMAD (Name & Roll No) 27600124195

ALI KHAN (Name & Roll No) 27600123004

BALARAM MANNA (Name & Roll No) 27600123013

ANUPAM PAYRA (Name & Roll No) 27600124197

Guide/Faculty: Prof. SAGAR CHAKRABORTY

Semester/Class: 5TH/3RD

Academic Year: 2025-26

Department of Computer Science & Engineering

Budge Budge Institute of Technology (BBIT)

Certificate

This is to certify that the mini project entitled "**Invoice Management System**" has been successfully completed by **GANESH KUMAR YADAV** (Name & Roll No) **27600123122** under the guidance of **Prof. SAGAR CHAKRABORTY**

at Department of Computer Science &Engineering, Budge Budge Institute of Technology, Kolkata

Guide/Faculty Name & Signature: _____

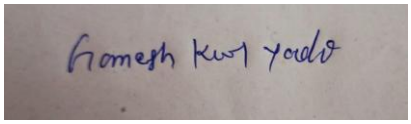
Head of Department: _____

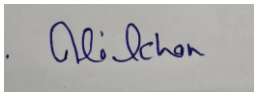
Date: _____ **Seal: _____**

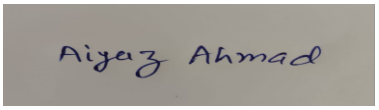
Acknowledgement

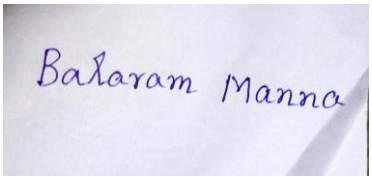
We would like to express our gratitude to all those who supported us in completing this mini project.

Signature of Team Members:









Abstract

This mini project focuses on the design and implementation of a simple **Invoice Management System** using the Java programming language. The problem addressed is the manual complexity of generating invoices that include customer details, purchased items, tax calculation, and a final summary. The objective of the project is to create an automated solution that allows users to enter customer information, add products or items, apply tax rules, and generate a formatted invoice efficiently.

The methodology adopted involves applying **Object-Oriented Programming (OOP)** principles such as abstraction, encapsulation, inheritance, and polymorphism. The system is structured into multiple classes, namely `Customer`, `Item`, `Invoice`, and `Main`, along with a `TaxCalculator` interface and its implementation `GSTTax`. This modular design ensures scalability, reusability, and ease of maintenance. User input is taken through the console, and the invoice is dynamically generated, displaying item details, subtotal, applicable tax, and grand total.

The tools used include **Java SE** for coding, **VS Code** as the development environment, and standard Java libraries for input/output operations.

The outcome of the project is a lightweight yet functional invoice generator that simplifies billing processes. It demonstrates how Java's OOP concepts and interface-based design can be used to solve real-world problems effectively while ensuring code flexibility for future enhancements, such as supporting multiple tax types or exporting invoices to files.

Table of Contents

Introduction

In today's fast-paced business environment, managing invoices accurately and efficiently has become an essential requirement for organizations of all sizes. Traditional manual billing processes are often time-consuming, prone to human errors, and inconvenient to maintain, especially when dealing with multiple customers and frequent transactions. An invoice not only serves as a payment record but also acts as an official financial document containing crucial details such as customer information, purchased items, applicable taxes, and the total payable amount. Therefore, a computerized invoice generation system offers a practical solution to reduce manual effort, improve accuracy, and streamline the billing workflow.

This project, titled **Invoice Management System**, is developed using the **Java programming language** with a strong emphasis on **Object-Oriented Programming (OOP) principles**. The system enables users to enter customer details, add items with quantity and price, calculate tax using an interface-driven design, and finally generate a neatly formatted invoice. By structuring the system into modular components such as `Customer`, `Item`, `Invoice`, and `TaxCalculator`, the project ensures flexibility, reusability, and ease of maintenance.

The project uses console-based user input to keep the implementation simple and interactive, while still reflecting real-world invoice generation steps. Additionally, the design supports extensibility, allowing new tax models or features to be integrated with minimal changes to existing code.

Beyond its academic purpose, this system demonstrates how fundamental programming concepts can be applied to develop practical applications. It also lays the foundation for more advanced billing solutions that could incorporate database integration, graphical user interfaces (GUIs), or export functionality for professional use in small businesses.

Problem Statement

In many small businesses and organizations, the process of generating invoices is still carried out manually using paper records or simple spreadsheets. While this approach may seem manageable for a limited number of transactions, it becomes inefficient and error-prone as the volume of customers and sales increases. Manual billing not only consumes extra time but also increases the chances of calculation mistakes, missing tax applications, and difficulty in maintaining proper records for future reference.

Furthermore, the lack of an automated system makes it challenging to ensure consistency and professionalism in invoice formatting. Businesses often require invoices that are clear, accurate, and standardized, both for customer satisfaction and for compliance with taxation policies. Without a systematic approach, tasks such as adding multiple items, applying different tax rates, or calculating totals can lead to delays and inaccuracies.

Therefore, there is a need for a simple yet reliable **Invoice Management System** that can automate the process of invoice creation. The system should allow entry of customer details, addition of items with quantity and price, automatic calculation of applicable tax, and generation of a formatted invoice. By addressing these challenges, the project aims to minimize manual effort, improve accuracy, and provide a flexible solution that can be extended in the future to meet advanced business needs.

Objectives

The primary objective of this project is to design and implement a simple, efficient, and user-friendly **Invoice Management System** using Java. The system aims to address the limitations of manual invoice preparation by automating the process and ensuring accuracy, reliability, and consistency in billing. The specific objectives of the project are:

1. **To collect and store customer details** such as name and address, ensuring that every invoice contains complete and accurate customer information.
2. **To allow the addition of multiple products/items** with their respective quantity and price, providing flexibility in handling different types of transactions.
3. **To apply tax calculations automatically** using an interface-based approach, ensuring modularity and allowing different tax models to be implemented easily.
4. **To generate a well-formatted invoice summary** that clearly displays customer details, list of items, subtotal, tax applied, and the final payable amount.
5. **To demonstrate the practical application of Object-Oriented Programming (OOP) principles** such as abstraction, encapsulation, and polymorphism in solving real-world problems.
6. **To ensure modularity and reusability of code** by organizing classes into separate packages (`model`, `invoice`, `tax`) for better structure and maintainability.
7. **To provide a foundation for future enhancements**, such as integration with databases, graphical user interfaces (GUIs), and file export features for professional use.

Through these objectives, the project seeks to simplify the billing process, reduce human error, and showcase how core Java concepts can be effectively applied to create real-time solutions.

Literature Survey / Existing System.

In the current business landscape, invoice generation is an essential task for maintaining financial records, ensuring proper customer communication, and complying with tax regulations. Traditionally, invoices were created manually on paper, which often led to errors in calculation, difficulty in record-keeping, and inefficiency in managing large numbers of transactions. With the advancement of digital technologies, many businesses have shifted toward using spreadsheets such as Microsoft Excel or Google Sheets for invoice creation. While these tools provide better accuracy compared to handwritten invoices, they still require significant manual effort and lack automation for repetitive tasks such as tax application and total calculation.

Existing commercial billing and accounting software, such as **Tally ERP**, **QuickBooks**, or **Zoho Invoice**, offer advanced features including customer management, inventory handling, automated tax calculation, and financial reporting. However, these systems may be complex and expensive for small-scale businesses or students who simply need a lightweight and easy-to-use solution. Additionally, such software often requires prior training and technical knowledge, which may not be feasible for all users.

The literature and existing systems highlight a gap for a simple, educational, and customizable solution that demonstrates the principles of invoice management without the overhead of full-scale accounting software. This project addresses that gap by developing a console-based **Invoice Management System** using Java. The proposed system emphasizes modularity through object-oriented design, enabling easy maintenance and extensibility. Unlike existing systems, it focuses on the core essentials—adding customer details, managing items, applying tax, and generating invoices—making it suitable as both a learning tool and a lightweight billing solution for small businesses.

System Requirements

11.1 Hardware Requirements

Minimum Requirements:

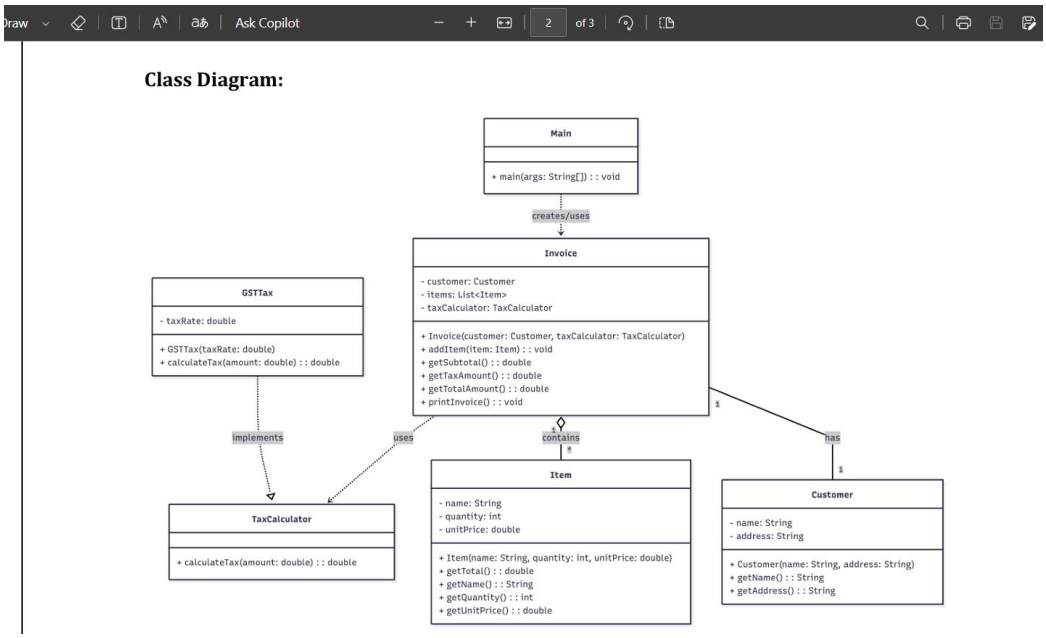
- Processor: Intel Core i3 (or equivalent)
- RAM: 2 GB
- Storage: 250 MB free disk space
- Display: 1024 × 768 resolution
- Input Devices: Standard keyboard and mouse

11.2 Software Requirements

- Processor: Intel Core i5 or above
- RAM: 4 GB or higher
- Storage: 500 MB free disk space
- Display: 1366 × 768 resolution or higher
- Input Devices: Standard keyboard and mouse

System Design

I



1. Customer Module (model.Customer)

- **Description:**
  This module handles customer-related information such as the customer's name and address.
- **Responsibilities:**
  - Store and retrieve customer details.
  - Provide data to the invoice for display.

---

2. Item Module (model.Item)

- **Description:**
  This module represents the products or items purchased by the customer. Each item includes a name, quantity, and unit price.
- **Responsibilities:**
  - Accept input for item details.
  - Compute the total price for each item (`quantity × unit price`).
  - Provide details for invoice printing.

---

3. Tax Module (tax.TaxCalculator & tax.GSTTax)

- **Description:**
  This module is responsible for applying tax to the subtotal. It is implemented using an interface (`TaxCalculator`) and a concrete class (`GSTTax`).
- **Responsibilities:**
  - Define the tax calculation behavior using the interface.
  - Allow flexibility for different types of tax in the future (e.g., Service Tax, VAT).
  - Calculate the tax amount based on the subtotal and tax rate.

---

4. Invoice Module (invoice.Invoice)

- **Description:**
  This is the core module responsible for integrating customer, items, and tax details to generate the final invoice.
- **Responsibilities:**
  - Maintain a list of items purchased by the customer.
  - Calculate subtotal, tax amount, and grand total.
  - Format and print the invoice in a structured manner.

---

5. Main Module (Main.java)

- **Description:**
  This module acts as the entry point of the application and coordinates between all other modules.
- **Responsibilities:**
  - Take input from the user (customer details, items, tax rate).
  - Create objects of different modules and pass data accordingly.
  - Call the invoice module to generate and display the final output.

Implementation

The implementation of the **Invoice Management System** was carried out in **Java** using a modular and object-oriented approach. The project is divided into separate packages—`model`, `invoice`, and `tax`—to ensure clean structure, code reusability, and maintainability.

1. Coding Approach

The project follows the principles of **Object-Oriented Programming (OOP)**:

- **Encapsulation**: Data such as customer details and items are stored in separate classes with private fields and public getters.
- **Abstraction**: The `TaxCalculator` interface defines the general behavior for tax calculation.
- **Polymorphism**: The system can support multiple types of tax calculators (e.g., GST, Service Tax) by implementing the interface differently.
- **Modularity**: Each functional unit (Customer, Item, Tax, Invoice) is implemented in its own class to ensure separation of concerns.

---

2. Algorithm Used

The system uses a **step-by-step sequential algorithm** for invoice generation:

1. **Input Phase:**
   - Accept customer details (name, address).
   - Accept item details (name, quantity, price).
   - Accept applicable tax rate.
2. **Processing Phase:**
   - Compute the subtotal by summing up the total of each item.
   - Apply the tax calculator to determine the tax amount.
   - Compute the grand total (`subtotal + tax`).
3. **Output Phase:**
   - Print a structured invoice including customer details, itemized purchases, subtotal, tax applied, and grand total.

This simple algorithm ensures accuracy and reduces manual calculation errors.

```java
public class InvoiceApp {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // 1. Enter customer details
        System.out.print(s:"Enter customer name: ");
        String name = sc.nextLine();
        System.out.print(s:"Enter customer address: ");
        String address = sc.nextLine();
        Customer customer = new Customer(name, address);

        // 2. Create Tax Calculator (e.g., GST @ 18%)
        TaxCalculator taxCalc = new GSTTax(rate:18);

        // 3. Create Invoice
        Invoice invoice = new Invoice(customer, taxCalc);

        // 4. Add products
        System.out.print(s:"Enter number of items: ");
        int n = sc.nextInt();
        sc.nextLine();
```

```java
interface TaxCalculator {
    double calculateTax(double amount);
}

// ----------------- GST Tax Implementation --------
class GSTTax implements TaxCalculator {
    private double rate;

    public GSTTax(double rate) {
        this.rate = rate;
    }

    @Override
    public double calculateTax(double amount) {
        return (amount * rate) / 100.0;
    }

    public double getRate() {
        return rate;
    }
}

// ----------------- Customer Class ----------------
```

Results & Discussion

Results

The **Invoice Management System** was successfully implemented and executed using Java. The system allows users to input customer details, add multiple items with quantity and price, specify the tax rate, and automatically generate a formatted invoice. The results from test runs show that the system is functioning as intended.

```
Enter customer name: GANESH YADAV
Enter customer address: BIHAR
Enter number of items: 2

Item 1
Product name: LAPTOP
Price: 55699
Quantity: 1

Item 2
Product name: MOUSE
Price: 1299
Quantity: 1

======= INVOICE =======
Customer: GANESH YADAV
Address : BIHAR
----------------------------
LAPTOP | Qty: 1 | Price: 55699.00 | Total: 55699.00
MOUSE | Qty: 1 | Price: 1299.00 | Total: 1299.00
----------------------------
Subtotal    : 56998.00
Tax (18%) : 10259.64
Grand Total: 67257.64
============================
```

Discussion

Performance:
The system runs smoothly on basic hardware since it is console-based and lightweight. Its modular design ensures efficiency even with multiple items, as subtotal and tax calculations are handled using Java's optimized libraries.

Accuracy:
By automating calculations, the system removes manual errors and generates clear, professional invoices with accurate subtotals, taxes, and totals.

Advantages:

- **Automation:** Fast and error-free invoice generation.
- **Flexibility:** Tax handled via an interface, allowing easy extension.
- **Maintainability:** Clean modular code (`model`, `invoice`, `tax`).
- **Portability:** Runs on any OS with Java support.

Conclusion

The **Invoice Management System** successfully demonstrates how Object-Oriented Programming principles can be applied to solve real-world problems like billing and invoice generation. The system automates customer detail entry, item management, tax calculation, and invoice formatting, ensuring efficiency, accuracy, and professionalism. Its modular design with separate packages (`model`, `invoice`, `tax`) makes the code easy to maintain, extend, and reuse.

Achievements:

- Automated generation of invoices with tax application.
- Accurate and error-free subtotal, tax, and total calculations.
- Console-based interaction that is simple and efficient.
- A scalable structure that can support future improvements.

Limitations:

- Currently, the system runs only as a console application with text-based input/output.
- No permanent storage of invoices (e.g., database or file system).
- Limited tax model support (only GST is implemented).

Future Enhancements:

- Integration with databases to store and retrieve invoices.
- Addition of more tax models, discounts, and promotional offers.
- A graphical user interface (GUI) for improved user experience.
- Export options such as PDF or Excel for professional use.
- Multi-user functionality for larger business operations.

References

1. Deitel, P. J., & Deitel, H. M. (2017). *Java: How to Program (10th ed.)*. Pearson Education.
2. Horstmann, C. S. (2019). *Core Java Volume I–Fundamentals (11th ed.)*. Prentice Hall.
3. Schildt, H. (2018). *Java: The Complete Reference (11th ed.)*. McGraw-Hill Education.
4. Oracle. (2023). *The Java™ Tutorials*. Retrieved from https://docs.oracle.com/javase/tutorial/
5. Baeldung. (2023). *Guide to Java Streams*. Retrieved from https://www.baeldung.com/java-streams
6. GeeksforGeeks. (2023). *Object-Oriented Programming in Java*. Retrieved from https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

A. Appendix

```java
class Customer {
    private String name;
    private String address;

    public Customer(String name, String address) {
        this.name = name;
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }
}
```

```java
package tax;

public interface TaxCalculator {
    double calculateTax(double amount);
    double getRate();
}
```

```java
class GSTTax implements TaxCalculator {
    private double rate;

    public GSTTax(double rate) {
        this.rate = rate;
    }

    @Override
    public double calculateTax(double amount) {
        return (amount * rate) / 100.0;
    }

    public double getRate() {
        return rate;
    }
}
```

B. User Manual

1. Run the program using any Java IDE (e.g., IntelliJ, Eclipse, NetBeans) or command line.
2. Enter customer details (name and address).
3. Enter applicable GST tax rate.
4. Enter the number of items to be added.
5. For each item, provide name, quantity, and unit price.
6. The system automatically generates and displays the invoice with subtotal, tax, and grand total.

B. Additional Diagrams

Workflow and Class Diagram