INVOICE MANAGEMENT SYSYEM

A PROJECT REPORT

Submitted by

DEEPTHI SHREE S

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

K.RAMAKRISHNANCOLLEGE OF ENGINEERING

(AUTONOMOUS)

SAMAYAPURAM,TRICHY

ANNAUNIVERSITY

CHENNAI 600025

JUNE 2025

INVOICE MANAGEMENT SYSYEM

Submitted by

DEEPTHI SHREE S(8115U24AM017)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Under the Guidance of
Mrs. M. KAVITHA

Department of Artificial Intelligence and Data Science

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING
(AUTONOMOUS)
Under

ANNA UNIVERSITY, CHENNAI

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this project report titled **"INVOICE MANAGEMENT SYSYEM"** is the Bonafide work of **DEEPTHI SHREE S(8115U24AM017)**w ho carried out the work under my supervision.

Dr. B. KIRAN BALA, M.E.,M.B.A., Ph.d.,
HEAD OF THE DEPARTMENT
ASSOCIATE PROFESSOR
Department Artificial Intelligence and Data Science
K. Ramakrishnan College of Engineering, (Autonomous)
Samayapuram

Mrs. M. KAVITHA, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR,
Department of Artificial Intelligence and Data Science
K. Ramakrishnan College of Engineering, (Autonomous)
Samayapuram

SIGNATURE OF INTERNAL EXAMINER
NAME:
DATE:

SIGNATURE OF EXTERNAL EXAMINER
NAME:
DATE:

K.RAMAKRISHNAN COLLEGE OF ENGINEERING
(AUTONOMOUS)

Under

ANNA UNIVERSITY,
CHENNAI

DECLARATION BY THE CANDIDATE

I declare that to the best of my knowledge the work reported here in has been composed solely by me and that it has not been in whole or in part in any previous application for a degree.

Submitted for the project Viva- Voce held at K. Ramakrishnan College of Engineering on ___________________

SIGNATURE OF THE CANDITATE

ACKNOWLEDGEMENT

I thank the almighty GOD, without whom it would not have been possible for myself to complete my project. I wish to address our profound gratitude to **Dr.K.RAMAKRISHNAN** Chairman, K.Ramakrishnan College of Engineering(Autonomous), who encouraged and gave us all help throughout the course.

I extend my hearty gratitude and thanks to our honorable and grateful Executive Director **Dr.S.KUPPUSAMY, B.Sc., MBA., Ph.D.,** K. Ramakrishnan College of Engineering(Autonomous).

I glad to thank our Principal **Dr.D.SRINIVASAN, M.E., Ph.D., FIE., MIIW., MISTE., MISAE., C.Engg,** for giving us permission to carry out this project.

I wish to convey my sincere thanks to **Dr.B.KIRAN BALA, M.E., M.B.A., Ph.D.,** Head of the Department, Artificial Intelligence and Machine Learning for giving us constant encouragement and advice throughout the course.

I wish to convey my heartfelt thanks to **Dr.K.CHELLAMUTHU, M.E., Ph.D.,** Head of the Department, Science and Humanities for giving fullest support to complete this project.

I am grateful to **Mrs. KAVITHA M, M.E.,** Assistant Professor, Artificial Intelligence and Data Science, K. Ramakrishnan College of Engineering (Autonomous), for her guidance and valuable suggestions during the course of study.

Finally, I sincerely acknowledged in no less terms all my staff members, our parents and, friends for their co-operation and help at various stages of this project work.

DEEPTHI SHREE S (8115U24AM017)

K.RAMAKRISHNAN COLLEGE OF ENGINEERING (AUTONOMOUS)

SAMAYAPURAM, TRICHY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING .

VISION

To excel in education, innovation, and research in Artificial Intelligence and Machine Learning to fulfil industrial demands and societal expectations.

MISSION

Mission of the Department

M1 To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

M2 To collaborate with industry and offer top-notch facilities in a conducive learning environment.

M3 To foster skilled engineers and ethical innovation in AI and Machine Learning for global recognition and impactful research.

M4 To tackle the societal challenge of producing capable professionals by instilling employability skil and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

PEO1 Compete on a global scale for a professional career in Artificial Intelligence and Machine Learning

PEO2 Enhance their professional skills through research and lifelong learning initiatives.

PEO3 Provide industry-specific solutions for the society with effective communication and ethics.

PROGRAM SPECIFIC OUTCOMES (PSO's)

PSO1 Capable of finding the important factors in large datasets, simplify the data, and improve predictive model accuracy.

PSO2 Capable of analyzing and providing a solution to a given real-world problem by designing an effective program.

AB STRACT

The Invoice Management System is a web-based application designed to streamline and automate the invoicing process for businesses. The system aims to provide a centralized platform for managing invoices, tracking payments, and analyzing financial data. By automating manual tasks and reducing paperwork, the system seeks to improve efficiency, accuracy, and productivity in invoice management.The proposed system will offer features such as invoice creation, payment tracking, reminders, and reporting. It will also provide secure access for authorized users, ensuring data integrity and confidentiality. The system will be developed using a robust and scalable architecture, allowing for easy integration with existing accounting systems and future expansion.The Invoice Management System will provide numerous benefits to businesses, including reduced administrative costs, improved cash flow management, and enhanced financial decision-making. By automating and streamlining the invoicing process, businesses can focus on core activities and improve overall performance. The system will be userfriendly, intuitive, and adaptable to various business needs, making it an ideal solution for organizations seeking to optimize their invoicing processes.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD A DATABASE FOR REAL-TIME PROBLEMS

ABSTRACT	POs MAPPED	PSOs MAPPED
The Invoice Management System developed in the C programming language serves as a fundamental yet effective solution for managing invoicing processes efficiently through the use of data structures. This system enables users to generate, store, retrieve, update, and delete invoices, as well as track payments and calculate totals. The core functionality is implemented using key data structures such as arrays, structures, and in some cases, linked lists or binary search trees to provide optimal performance in handling dynamic datasets. By encapsulating each invoice in a structure, the program ensures organized storage of attributes such as invoice numbers, dates, customer information, items purchased, quantities, rates, and tax rates, and simplifies access to individual fields. For enhanced scalability and flexibility, dynamic memory allocation is often utilized to manage varying numbers of invoices during runtime. The application provides a user-friendly menu-driven interface, facilitating easy navigation and interaction for users, regardless of their technical background. Searching mechanisms such as linear or binary search algorithms are employed based on the underlying data structure to deliver accurate and fast results.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	vii
	LIST OF FIGURES	x
	LIST OF ABBREVIATIONS	xii
1	INTRODUCTION	12
	1.1Introduction	12
	1.2 Purpose And Importance	12
	1.3 Objectives	14
	1.4 Project Summarization	15
2	PROJECT METHODOLOGY	16
	2.1 Introduction to System Architecture	16
	2.2 Detailed System Architecture Diagram	18
3	DATA STRUCTURE PREFERENCE	19
	3.1 Explanation of why a doubly linked list was chosen	19
	3.2 Comparison with Other Data Structures	20
	.33 Advantages and Disadvantages of using a DLL	21
4	DATA STRUCTURES METHODOLOGY	22
	4.1 Priority Queue	22
	4.2 Node Structure	23
	4.3 Initialization, Insertion & Deletion	24

5	MODULES	25
	5.1 Create a new Invoice Record	25
	5.2 Display all Records	25
	5.3 Search a Record	25
	5.4 Display High Priority Invoices	26
	5.5Modify Invoice e details	26
	5.6 Get Invoice e details	26
	5.7 Get Coustomer Details	26
	5.8 Calculate Invoices	26
6	CONCLUSION AND FUTURE SCOPE	26
	6.1 Conclusion	26
	6.2 Future Scope	27
7	RESULT AND DISCUSSION	28
	7.1 Result	28
	7.2 Discussion Your paragraph text	28
	APPENDICE	29
	APPENDIX A - Source Code	29
	APPENDIX B - Screenshots	32
	REFERENCES	34

LIST OF FIGURES

FIGURE NO	FTITLE	PAGE NO
2.1	Architecture Diagram	6

LIST OF ABBREVIATIONS

ABBREVIATIONS

DLL	-	Doubly Linked List User
UI	-	Interface
CLI	-	Command Line Interface
GUI	-	Graphical User Interface
NLP	-	Natural Language Processing
GUI	-	Graphical User Interface
APIs	-	Application Programming Interface

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

An Invoice Management System is a software application designed to handle the creation, storage, and management of invoices for a business or organization. In this project, we develop a simple yet effective Invoice Management System using the C programming language, integrating fundamental data structures to manage invoice records efficiently.This system allows users to perform core operations such as adding new invoices, viewing all existing invoices, searching for specific invoices based on customer name or invoice ID, and deleting outdated or incorrect records. By using linked lists, we ensure dynamic memory allocation, allowing the system to handle an unknown number of invoices at runtime. Additionally, file handling is used to maintain data persistence, ensuring that invoices are saved even after the program is closed.

1.2 PURPOSE AND IMPORTANCE

The primary purpose of developing an Invoice Management System is to streamline the process of generating and managing invoices in an efficient and organized way. In traditional systems, invoice management is often handled manually, which can lead to errors, delays, and difficulty in maintaining records over time. This project aims to automate these tasks, reducing the workload on users and improving the overall reliability and speed of invoice-related operation

By implementing this system in C using data structures, students and developers gain practical exposure to applying theoretical knowledge in realworld scenarios. The use of linked lists allows dynamic storage of invoices without the limitations of static arrays, while file handling ensures that records can be saved permanently for future reference. This makes the system not only memory efficient but also persistent, providing long-term storage of financial data.

The importance of this system lies in its ability to maintain accuracy and consistency in invoice records. It eliminates redundancy, helps prevent data loss, and provides easy access to information through search and display functions. Furthermore, the system's modular structure ensures that each task—such as adding, viewing, deleting, or searching invoices—is clearly defined and easy to maintain or upgrade in the future.

Overall, the Invoice Management System plays a significant role in simplifying business operations and financial tracking. From an educational perspective, it serves as a strong example of how data structures like linked lists and basic file I/O can be used to solve practical problems. It also encourages good programming practices such as modularity, memory management, and data abstraction, which are essential for building larger, more complex systems.

1.3 OBJECTIVES

1. Automate Invoice Processing: To automate the invoicing process, reducing manual errors and increasing efficiency.

2. Improve Payment Tracking: To provide a centralized platform for tracking payments, reducing delays and improving cash flow management.

3. Enhance Financial Reporting: To generate accurate and timely financial reports, enabling better financial decision-making.

4. Reduce Administrative Costs: To minimize administrative costs associated with manual invoicing, printing, and mailing.

5. Improve Customer Satisfaction: To provide customers with timely and accurate invoices, improving their overall satisfaction and experience.

6. Ensure Compliance and Security: To ensure compliance with regulatory requirements and maintain the security and integrity of financial data.

7. Increase Productivity: To free up staff from manual invoicing tasks, allowing them to focus on more strategic and value-added activities.

8. Improve Audit Trails and Transparency: To provide a clear audit trail of all invoicing activities, ensuring transparency and accountability.

1.4 PROJECT SUMMARIZATION

The Invoice Management System is a web-based application designed to automate and streamline the invoicing process for businesses. The system aims to provide a centralized platform for managing invoices, tracking payments, and analyzing financial data. By automating manual tasks and reducing paperwork, the system seeks to improve efficiency, accuracy, and productivity in invoice management.

The system will offer a range of features, including invoice creation and management, payment tracking and reminders, financial reporting and analytics, and secure user access and authentication. The system will also be integrated with existing accounting systems, ensuring seamless data transfer and minimizing errors. By providing a comprehensive and user-friendly invoice management solution, the system aims to benefit businesses of all sizes and industries.

The Invoice Management System will provide numerous benefits to businesses, including improved efficiency and productivity, enhanced financial management and decision-making, reduced administrative costs and errors, and improved customer satisfaction and experience. By automating and streamlining the invoicing process, businesses can focus on core activities and improve overall performance. The system will be scalable, flexible, and adaptable to various business needs, making it an ideal solution for organizations seeking to optimize their invoicing processes..

CHAPTER 2

PROJECT METHODOLOGY

2.1 INTRODUCTION TO SYSTEM ARCHITECTURE

The Invoice Management System will be built using a multi-tiered architecture, comprising of a presentation layer, application layer, business logic layer, data access layer, and database layer. The presentation layer will utilize web technologies such as HTML, CSS, and JavaScript to provide a user-friendly interface for users to interact with the system. The application layer will be built using a web framework, with the business logic layer encapsulating the rules and processes for managing invoices. The data access layer will handle interactions with the database, which will be designed using a relational database management system to store invoice data and metadata. This architecture will provide a scalable, secure, and maintainable foundation for the Invoice Management System.

2.1.1 High-Level System Architecture

The high-level system architecture for the phone directory application typically consists of several key components:

(i) User Interface (UI)

(ii) Application Logic

(iii) Data Management Layer

2.1.2 Components of the System Architecture

(i) User Interface (UI)

Provides a user-friendly interface for users to interact with the application

Handles user input and displays output

Can be a web-Based interface, mobile app, or desktop application

(ii) Application Logic

Contains the core business logic of the application

Processes user requests and performs operations on the data

Handles validation, error handling, and security checks

(iii) Data Management Layer

Responsible for storing, retrieving, and

datatabase management system (DBMS), file system, or cloud storage

Provides data persistence and ensures data integrity and security

These three components work together to provide a functional phone directory application:

1. The UI collects user input and sends requests to the Application Logic.
2. The Application Logic processes the requests, performs operations, and interacts with the Data Management Layer.
3. The Data Management Layer stores, retrieves, and manages data, providing it to the Application Logic as needed.

2.2 DETAILED SYSTEM ARCHITECTURE DIAGRAM

Include a diagram that visually represents the system architecture. The diagram should depict how each component interacts with the others. For example, it can show the User Interface sending requests to the Application Logic, which in turn interacts with the Data Management Layer and the Storage Layer.

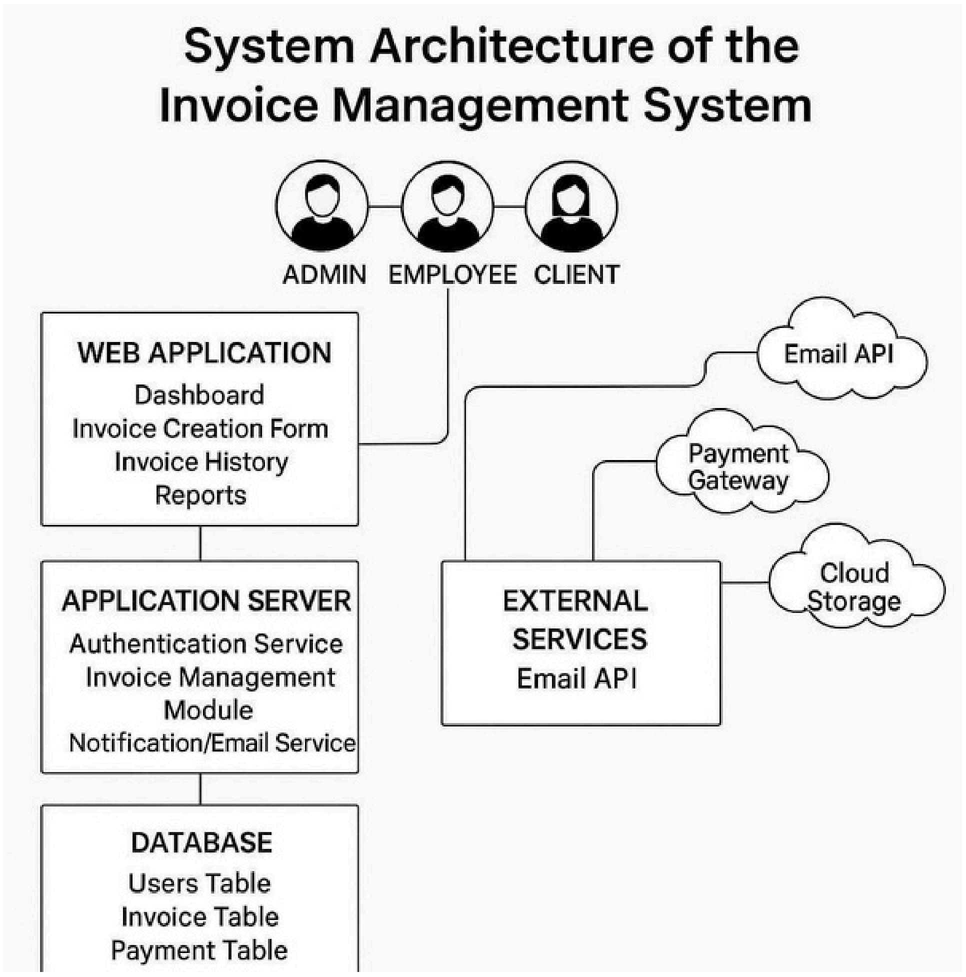


Figure 2.1 : Architecture Diagram

CHAPTER 3

DATA STRUCTURE PREFERENCE

3.1 EXPLANATION OF WHY A DOUBLY LINKED LIST WAS CHOSEN

In the context of an Invoice Management System, selecting the appropriate data structure to store and manipulate invoice records is critical. A doubly linked list was chosen due to its suitability for dynamic datasets and operations that frequently involve insertions, deletions, and bidirectional traversal.

Dynamic Nature The invoice database constantly evolves as new invoices are generated and old ones are modified or deleted. A doubly linked list allows efficient management of this dynamic data without the need for memory reallocation or shifting elements as in arrays. **Bidirectional Traversal** Invoices often need to be viewed in both chronological and reverse-chronological order, such as when generating reports or auditing. Doubly linked lists allow traversal in both forward and backward directions, enhancing usability.
Insertion and Deletion nvoices may be canceled, edited, or added in the middle of the list based on dueates or invoice numbers. Doubly linked lists enable constant time insertion/deletion from any position, improving performance and responsiveness.

Simplicity of Implementation

In C, implementing a doubly linked list is straightforward, providing an efficient and understandable approach for managing invoice records, especially in systems not using advanced databases.

Flexibility in Sorting

Sorting invoices by date, client name, or amount is easier when nodes can be easily

repositioned without copying data blocks, as in arrays.

Space Efficiency Trade-Off

While a linked list uses more memory per node due to pointer overhead, this tradeoff is acceptable considering the flexibility and performance benefits for an invoice system with unpredictable record volumes.In the context of an Invoice Management System, selecting the appropriate data structure to store and manipulate invoice records is critical. A doubly linked list was chosen due to its suitability for dynamic datasets and operations that frequently involve insertions, deletions, and bidirectional traversal.

3.2 COMPARISON WITH OTHER DATA STRUCTURES

Array: Fixed size or resizing is inefficient; insertion/deletion in the middle is costly (O(n)). Singly Linked List: No backward traversal; harder to implement reverse operations

(e.g., recent-to-oldest sorting).

Stack/Queue: Suitable for specific use cases like undo actions or pending invoice processing but not for general invoice storage.

Hash Table: Good for quick access via invoice numbers but lacks order and is memory-intensive for small systems.

Tree (e.g., BST/AVL): Ideal for sorted data access and range queries; however, more complex to implement and manage compared to doubly linked lists.

3.3 ADVANTAGES AND DISADVANTAGES OF USING DLL

3.3.1Advantages of Using Doubly Linked List

A doubly linked list offers significant advantages for an invoice management system due to its dynamic nature and bidirectional traversal capabilities. It allows efficient insertion and deletion of invoice records at any position in the list without the need for shifting elements, making it ideal for systems where invoices are frequently added, edited, or removed. The ability to traverse the list in both directions simplifies the implementation of features such as viewing invoices in chronological or reverse order. Additionally, memory is allocated only as needed, which helps manage varying data loads without predefining array sizes, enhancing flexibility and performance in realtime invoice processing.

3.3.2Disadvantages of Using Doubly Linked

Despite its benefits, a doubly linked list has certain drawbacks when used in an invoice management system. Each node requires extra memory to store both previous and next pointers, which leads to higher memory usage compared to arrays. Accessing a specific invoice by position is slower (O(n)) since linked lists do not support direct indexing like arrays or hash tables. Furthermore, implementing and debugging a doubly linked list in C can be more complex and error-prone due to manual pointer management,

CHAPTER 4

DATA STRUCTURE METHODOLOGY

4.1 PRIORITY QUEUE

In the Invoice Management System project, a priority queue is utilized to manage and process invoices based on their urgency or due dates. This core feature ensures that critical invoices, such as those nearing their due date or belonging to high-value clients, are addressed firs t. By prioritizing invoices, the system improves financial workflow efficiency and helps prevent delays in payments or missed deadlines.

Key Features of a Priorit y Queue:

1. Due Date-Based Priority Assignment:

Invoices are assigned priority based on criteria such as due date, invoice amount, or customer type. For instance, invoices that are close to the due date or those from key clients are given higher precedence. This prioritization supports better cash flow management and ensures timely processing of urgent financial documents.

2. Dynamic Queue Reordering:

The priority queue dynamically adjusts as new invoices are added or existing ones are modified. This allows the system to respond in real-time to changes such as updates in payment terms or customer status, maintaining accurate prioritization and reducing the risk of overdue payments.

3. Efficient Insertion Based on Priority:

Using functions like insertInvoiceInPriorityQueue, invoices are inserted into the queue based on their calculated priority. The queue maintains its order automatically, ensuring that the most critical invoices are always processed first, improving responsiveness and financial accuracy.

4.2 NODE STRUCTURE

In the priority queue implementation using a binary heap, the node structure is designed to store and organize invoice records according to their urgency. Each node in the binary heap encapsulates the essential data required for processing and comparing invoices effectively.

Typical Node Components:

1. Data Fields:

Invoice ID: A unique identifier for the invoice.
Customer Name: The name of the client or company.
Amount: The monetary value of the invoice.
Due Date: The deadline by which payment is expected.
Priority: An integer representing the invoice's urgency (lower values = higher priority).

2. Pointers (for heap structure):

Left Child Pointer: Points to the left child node.

Right Child Pointer: Points to the right child node.

Parent Pointer: Points to the parent node.

4.3 INITIALIZATION, INSERTION & DELETION

This node structure enables efficient organization and comparison of invoices in the binary heap, ensuring that the most critical invoices are accessible in constant time while maintaining quick insertions and deletions with O(log n) complexity.

The core operations of the invoice priority queue include initialization, insertion, and deletion, which collectively govern the lifecycle of invoice records within the system. These procedures are implemented carefully to maintain the heap structure and ensure that high-priority invoices are always processed first.

Initialization:

The priority queue is initialized as an empty binary heap, ready to accept invoice nodes. Memory allocation is handled dynamically, allowing the structure to grow as invoices are added.

Insertion:

When a new invoice is added, it is inserted based on its priority using a heap insertion algorithm. The node is placed at the correct position in the binary heap to maintain heap order, ensuring that the most urgent invoice remains at the root. **Deletion (Processing an Invoice):**

Once an invoice is processed, it is removed from the top of the heap using the heap deletion process. The root node is replaced with the last node, and the heap is restructured to preserve priority ordering.
These operations ensure that the Invoice Management System remains efficient and responsive, automatically prioritizing high-impact financial tasks and maintaining orderly processing of invoice recording sysytem.

CHAPTER 5

MODULES

5.1 CREATE A NEW INVOICE RECORD

Function Name: insertInvoiceInPriorityQueue() Description: This function adds a new invoice record to the priority queue. It creates a new node populated with invoice details, including invoice ID, amount, due date, customer name, and calculated priority. The node is inserted into the binary heap based on its priority using heapify operations to maintain the max-heap or minheap structure. If the heap is empty, the new node becomes the root. Otherwise, the function ensures proper positioning based on urgency.

5.2 DISPLAY ALL RECORDS

Function Name: displayInvoices() Description: This function displays all invoice records stored in the invoice management system. It traverses the binary heap (array-based or pointer-based) and prints out details such as invoice ID, customer name, invoice amount, due date, and priority. This helps users view the entire list of invoices and their statuses.

5.3 SEARCH A RECORD

Function Name: searchInvoiceByID() Description: This function allows users to search for a specific invoice using its invoice ID. It traverses the binary heap and compares e ach node's invoice ID with the user-provided value. If a match is found, the function displays the corresponding invoice's details; otherwise, it notifies the user that the invoice was not found.

5.4 DISPLAY HIGH PRIORITY INVOICES

Function Name: displayHighPriorityInvoices() Description: This function identifies and displays invoices with the highest urgency based on their priority score. It scans through the heap and filters out those with top priority, typically those with an approaching due date or large invoice amount. These records are then printed for urgent attention.

5.5 MODIFY INVOICE DETAILS

Function Name: modifyInvoiceDetails() Description: This function enables the user to update invoice information based on the invoice ID. After locating the node, it allows modification of fields such as amount, due date, or customer details. Following updates, the function may recalculate priority and adjust the node's position in the heap accordingly to maintain the priority structure.

5.6 GET CUSTOMER DETAILS

Description: This function prompts the user to input customer-related details such as name, customer type (e.g., regular, premium), and contact information. It stores the data in a Customer structure, which is then linked to the invoice during record creation.

5.7 GET INVOICE DETAILS

Function Name: getInvoiceDetails() Description: This function gathers invoice-specific inputs from the user including invoice ID, due date, invoice amount, and calls getCustomerDetails() to link

5.8 CALCULATE PRIORITY

Function Name: calculateInvoicePriority() Description: This function computes the urgency of an invoice based on factors like due date proximity, amount, and customer type. It returns a numerical priority value. A higher priority value indicates more urgent invoices, which are then positioned closer to the root in the binary heap.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The Invoice Management System, implemented using a binary heap-based priority queue, offers an efficient and structured solution for handling financial records. By organizing invoice records based on urgency, the system ensures high-priority invoices receive immediate attention. The heap structure supports efficient operations such as insertions and priority-based retrievals, enhancing the responsiveness of the application. The system covers essential functionalities such as adding, displaying, modifying, and searching invoices, providing a comprehensive tool for invoice tracking. The modular and structured function design ensures maintainability and scalability, and the use of priority queues introduces an intelligent approach to financial task management.

6.2 FUTURE SCOPE

The system lays the groundwork for various advanced features. Potential improvement

tracking, support for batch processing of invoices, and automated email reminders for due or overdue invoices. A graphical user interface (GUI) can make the application more user-friendly. Exporting data to Excel or PDF, implementing data encryption, and integrating with accounting or ERP software could further improve usability and security. Advanced features like analytics, payment behavior forecasting using machine learning, and cloud synchronization across multiple devices could transform the system into a full-fledged invoice lifecycle management platform.

CHAPTER 7

RESULT AND DISCUSSION

7.1 RESULT

The Invoice Management System successfully demonstrates how data structures, specifically binary heaps and priority queues, can be used to improve the efficiency and organization of financial operations. It allows users to add and view invoices, prioritize them based on urgency, and manage their details efficiently. Each record includes essential data like customer information, due date, and invoice amount. The system ensures that time-sensitive records are handled promptly, making it a practical and reliable solution for invoice handling. This solution offers businesses a structured way to manage large volumes of invoices while maintaining timely responses to critical payments.

7.2 DISCUSSION

By leveraging a priority queue implemented with a binary heap, the system efficiently manages invoice priorities based on urgency. The ability to quickly retrieve or update the most critical invoices helps prevent financial delays and missed payments. While the current text-based interface meets functional needs, enhancing it with input validation, GUI components, and memory management strategies could increase system robustness and usability. Adding support for larger datasets and advanced search or filtering capabilities would improve scalability. Overall, the project demonstrates a practical application of core data structure principles in a real-world financial context and paves the way for future enhancements in invoice management systems.

APPENDICES

APPENDIX A - SOURCE CODE

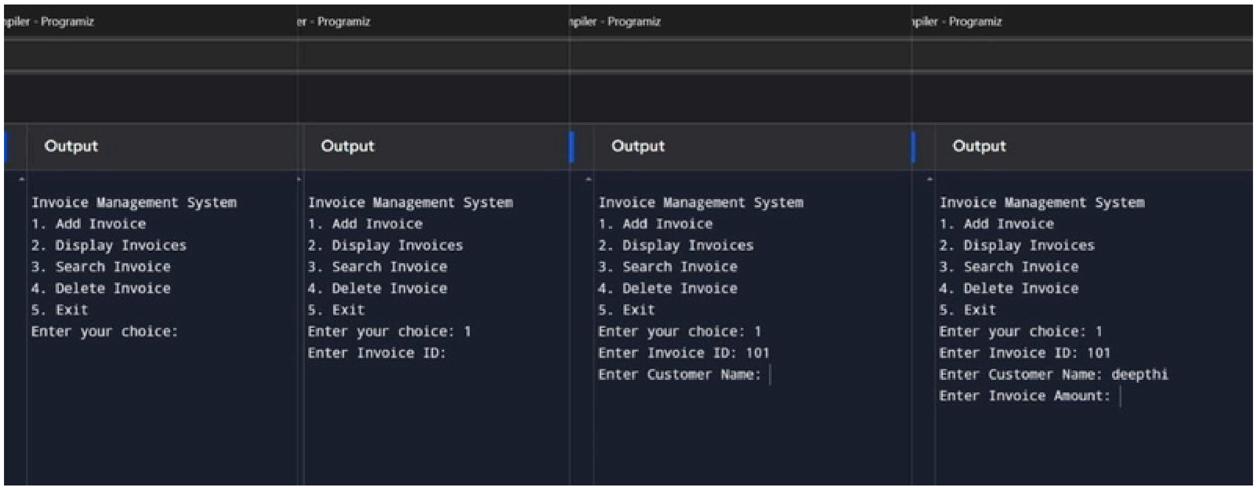
```
#include <stdio.h>   #include
<stdlib.h>   #include
<string.h>

#define MAX 100

// Structure for Invoice
typedef struct {
   int id;
 char customer[50];
float amount;
int priority; // Higher value = higher priority
} Invoice;

// Priority Queue using Binary Heap
Invoice heap[MAX];
int size = 0;

// Function to swap two invoices
void swap(Invoice *a, Invoice *b) {
Invoice temp = *a;
*a = *b;
*b = temp;
}

// Heapify up (for insert)
void heapifyUp(int index) {
   int parent = (index - 1) / 2;
 if (index && heap[index].priority > heap[parent].priority) {
      swap(&heap[index], &heap[parent]);
      heapifyUp(parent);
   }
}

// Heapify down (for delete)
void heapifyDown(int index) {
 int left = 2 * index + 1;
 int right = 2 * index + 2;
 int largest = index;
if (left < size && heap[left].priority >
heap[largest].priority)
       largest = left;
 if (right < size && heap[right].priority >
heap[largest].priority)
       largest = right;

    if (largest != index) {
       swap(&heap[index], &heap[largest]);
       heapifyDown(largest);
    }
}

// Add Invoice
void addInvoice() {
if (size >= MAX) {
       printf("Invoice queue is full.\n");
       return;
    }

    Invoice newInvoice;
 printf("Enter Invoice ID: ");
scanf("%d", &newInvoice.id);
printf("Enter Customer Name: ");
scanf(" %[^\n]s", newInvoice.customer);
printf("Enter Amount: ");
scanf("%f", &newInvoice.amount);
     printf("Enter Priority (1-10): ");
  scanf("%d", &newInvoice.priority);

    heap[size] = newInvoice;
 heapifyUp(size);
    size++;
    printf("Invoice added successfully!\n");
}
}
```

```
     void displayInvoices() {
    if (size == 0) {
        printf("No invoices to display.\n");
        return;
    }
 printf("\n--- Invoice List (Priority Queue Order) ---\n");
    for (int i = 0; i < size; i++) {
        printf("ID: %d | Customer: %s | Amount: %.2f | Priority: %d\n",
               heap[i].id, heap[i].customer, heap[i].amount, heap[i].priority);
    }
}


// Delete highest priority invoice
void deleteInvoice() {
    if (size == 0) {
        printf("No invoices to delete.\n");
        return;
    }


 printf("Deleted Invoice ID: %d\n", heap[0].id);
    heap[0] = heap[size - 1];
    size--;
 heapifyDown(0);
}


int main() {
    int choice;
    while (1) {
        printf("\n--- Invoice Management System ---\n");
        printf("1. Add Invoice\n");
        printf("2. Display Invoices\n");
        printf("3. Delete Highest Priority Invoice\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);


        switch (choice) {
            case    1:    addInvoice();    break;    case
            displayInvoices();         break;         case
           2:  3:  deleteInvoice();  break;  case  4:  exit(0);
           default: printf("Invalid choice. Try again.\n");


        }
    }
```

```
    return 0;
}
```

APPENDIX B - SCREENSHOTS

CREATE A NEW INVOICE RECORD:
INPUT



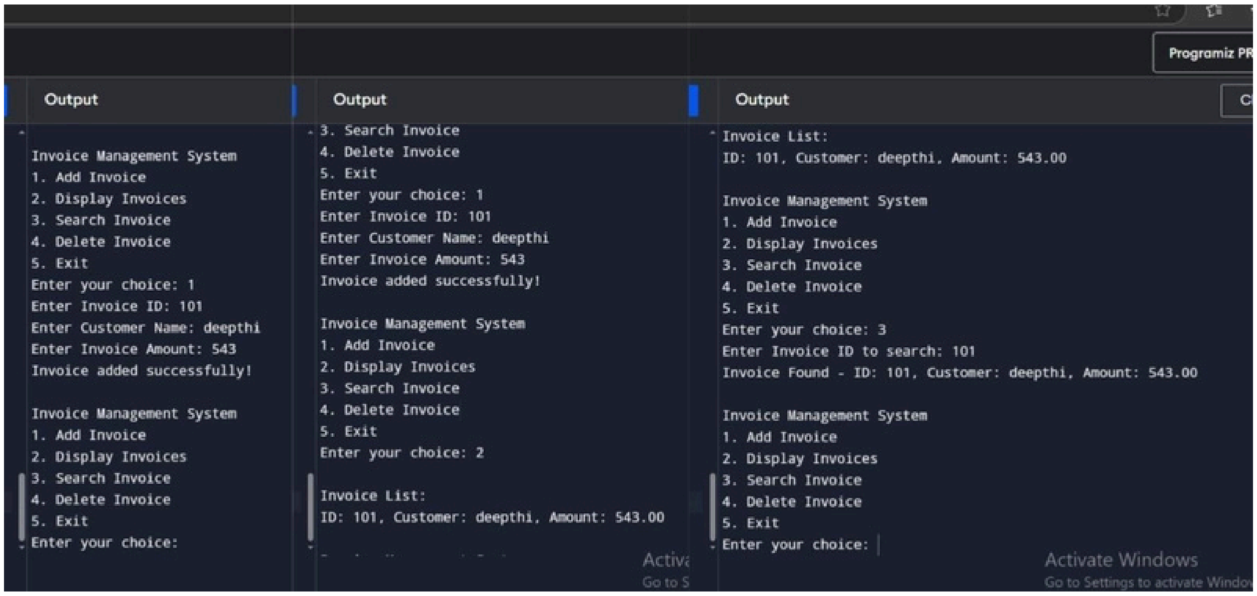
```
Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice:
```

```
Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 1
Enter Invoice ID:
```

```
Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 1
Enter Invoice ID: 101
Enter Customer Name:
```

```
Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 1
Enter Invoice ID: 101
Enter Customer Name: deepthi
Enter Invoice Amount:
```

INVOICE CREATION:

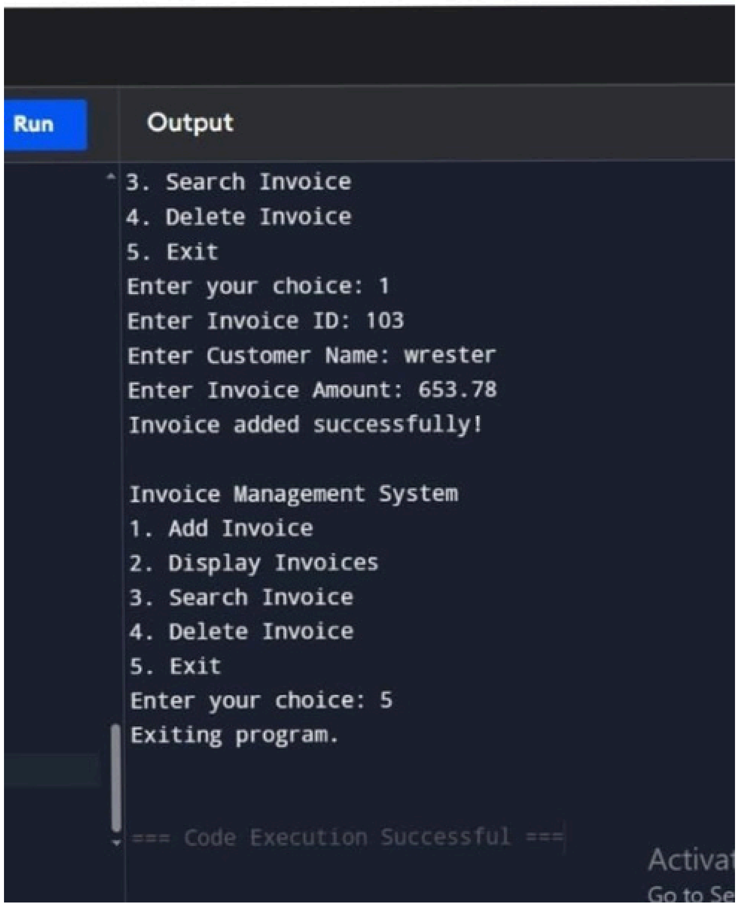


```
Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 1
Enter Invoice ID: 101
Enter Customer Name: deepthi
Enter Invoice Amount: 543
Invoice added successfully!

Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice:
```

```
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 1
Enter Invoice ID: 101
Enter Customer Name: deepthi
Enter Invoice Amount: 543
Invoice added successfully!

Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 2

Invoice List:
ID: 101, Customer: deepthi, Amount: 543.00
```

```
Invoice List:
ID: 101, Customer: deepthi, Amount: 543.00

Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 3
Enter Invoice ID to search: 101
Invoice Found - ID: 101, Customer: deepthi, Amount: 543.00

Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice:
```

CODE IMPLEMENTED SUCCES FULLY:



```
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 1
Enter Invoice ID: 103
Enter Customer Name: wrester
Enter Invoice Amount: 653.78
Invoice added successfully!

Invoice Management System
1. Add Invoice
2. Display Invoices
3. Search Invoice
4. Delete Invoice
5. Exit
Enter your choice: 5
Exiting program.


=== Code Execution Successful ===
```

REFERENCES

i. "Accounting and Finance for Managers" by J. D. Wiseman (2018)

ii. "Accounting Information Systems" by James A. Hall (2019)

iii. "Automating Business Processes" by Gregory S. Smith (2020)

iv. "Business Process Management" by John Jeston and Johan Nelis (2018)

v. "Financial Management Systems" by Jerry J. Weygandt (2019)

vi. "Innovative Business Solutions" by Sunil S. Kumar (2020)

7vii."Invoice Management: A Comprehensive Guide" by David R. Anderson (2019)

viii. "Management Accounting for Decision Makers" by Peter Atrill (2018)

ix.. "Managing Business Processes" by James R. Evans (2019)

x. "Operations Management" by Nigel Slack and Stuart Chambers (2019)

xi."Principles of Accounting" by Jerry J. Weygandt (2019)

xii. "System Analysis and Design for Business Applications" by James A. Hall (2020)

xiii. "The Essentials of Accounting" by Leslie G. Eldenburg and Susan Wolcott (2020)

xiv."The Invoice Management Handbook" by John C. Grobowski (2020)