K L UNIVERSITY

FRESHMAN ENGINEERING DEPARTMENT

A Project Based Lab Report

On

BILL NOTIFICATION SYSTEM

SUBMITTED BY

2400090052 G.NIHAL NAGA SAI

UNDER THE ESTEEMED GUIDANCE OF

Mr. BELLAM SURENDRA BABU

Assistant Professor

KL UNIVERSITY

Green fields, Vaddeswaram – 522 502

Guntur Dt., AP, India.

DEPARTMENT OF BASIC ENGINEERING SCIENCES

CERTIFICATE

This is to certify that the project based laboratory report entitled "**BILL NOTIFICATION SYSTEM**" submitted by G.NIHAL NAGA SAI bearing Regd. No. **2400090052** to the Department of Basic Engineering Sciences, KL University in partial fulfillment of the requirements for the completion of a project in "***Data Structures- 24SC1203***" course in I B Tech II Semester, is a Bonafide record of the work carried out by him under my supervision during the academic year 2024-25.

PROJECT SUPERVISOR — **HEAD OF THE DEPARTMENT**

BELLAM SURENDRA BABU — Dr. D. HARITHA

ACKNOWLEDGEMENTS

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project-based laboratory report.

I express the sincere gratitude to our director **Dr. A Jagadeesh** for his administration towards our academic growth.

I express sincere gratitude to our principal **Dr. V. Krishna Reddy** and HOD-BES **Dr. D. Haritha** for her leadership and constant motivation provided in successful completion of our academic semester. I record it as my privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to our project supervisor **B Surendra Babu** for his novel association of ideas, encouragement, appreciation, and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

Name: G.NIHAL NAGA SAI

Regd. No: 2400090052

ABSTRACT

The Bill Notification System is designed to automate the process of tracking and notifying users about upcoming bill payments, reducing the chances of missed deadlines and late fees. This system employs fundamental data structures such as **queues**, **linked lists**, and **hash tables** to manage and organize user data, bill records, and notification schedules efficiently.

User information and bill details are stored using **hash tables** for quick access and updates. A **priority queue** is implemented to manage bill due dates, ensuring that the most urgent bills are notified first. Additionally, **linked lists** help maintain dynamic lists of recurring bills for each user. The notification mechanism is triggered by comparing the current date with due dates, automatically sending alerts via email or SMS.

By leveraging these core data structures, the system ensures optimal performance, scalability, and reliability in delivering timely reminders. This approach minimizes manual effort and enhances user convenience in managing multiple financial obligations.

INDEX

INTRODUCTION

➢ **Objectives:**

1. Automate Bill Tracking: To automatically store and manage multiple bill types for each user, including due dates and payment amounts.
2. **Timely Notifications:** To send timely alerts or reminders to users via SMS, email, or app notification prior to bill due dates.
3. **Efficient Data Handling:** To use optimal data structures (hash tables, queues, linked lists, etc.) for efficient storage, retrieval, and update of billing information.
4. **User-Friendly Interface:** To provide an intuitive and interactive user interface for users to view, edit, and add bills.
5. **Scalability:** To ensure the system can handle a growing number of users and bills without performance degradation.

➢ **Data Structures Used:**

Component	**Data Structure**	**Purpose**
User Profiles	Hash Table	Quick access to user data
Bill List per User	Linked List	Dynamic addition/removal of bills
Notification Scheduler	Priority Queue	Order bills by urgency (due date)
Notification Log	Queue	First-in, first-out delivery tracking

➢ **Data Flow Diagram (DFD) - Level 1 (Textual Description):**

1. **User → [Bill Notification System]:** User submits bill details (amount, due date, etc.).
2. **[Bill Notification System] → [Bill Storage]:** System stores the data using linked lists and hash tables.
3. **[Scheduler] → [Bill Storage]:** Periodically checks for bills due soon using a priority queue.
4. **[Scheduler] → [Notification System]:** Sends notification requests.
5. **[Notification System] → [User]:** User receives alert via selected channel.

AIM

To develop a **Bill Notification System** that utilizes efficient **data structures** such as hash tables, linked lists, and priority queues to store, manage, and process user bill information, enabling timely notifications and ensuring optimal performance in bill tracking and reminder generation.

ADVANTAGES: -

1.Efficient Data Management:

Using hash tables and linked lists ensures quick access, insertion, and deletion of user and bill data.

2.Timely Notifications:

With priority queues, bills are sorted by due date, allowing the system to prioritize and send alerts for the most urgent bills first.

3.Reduced Missed Payments:

Automated reminders help users avoid late fees, service disconnections, or penalties.

DISADVANTAGES: -

1.Complexity in Implementation:

Using multiple data structures (e.g., hash tables, priority queues) may increase the system's design and coding complexity, especially for beginners.

2.Memory Consumption:

Data structures like hash tables and linked lists may consume more memory, especially when handling a large number of users and bills.

3.Data Consistency Issues:

Without proper synchronization and error handling, inconsistencies may occur during updates or deletions in dynamic structures.

FUTURE IMPLEMENTATION: - In the future, the Bill Notification System can be enhanced with online payment integration, cloud storage for scalability, and mobile app support for better accessibility. Machine learning can be used to predict bills and spending patterns. Features like calendar sync, voice assistant support, and multilingual interfaces can improve user experience. Advanced security measures and IoT integration can also be added to ensure safety and real-time updates.

SYSTEM REQUIREMENTS

➢ SOFTWARE REQUIREMENTS:

The major software requirements of the project are as follows:

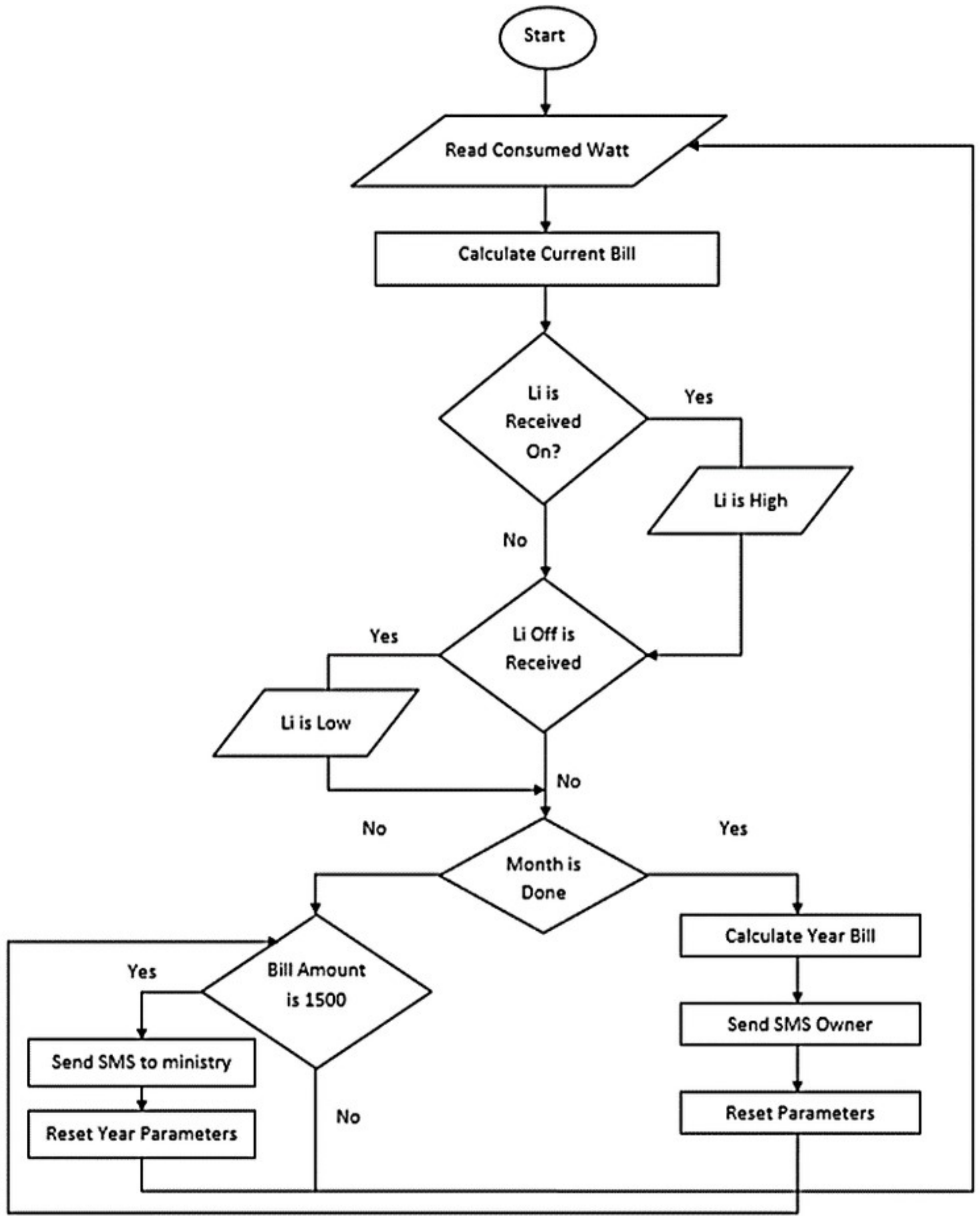
Language	:	C-language
Operating system	:	Windows 8 or later.
Tools	:	DEV C++ IDE

➢ HARDWARE REQUIREMENTS:

The hardware requirements that map towards the software are as follows:

RAM	: 2GB or later
Processor	: intel i3 or later

FLOWCHART


Start
Read Consumed Watt
Calculate Current Bill
Li is Received On?
Yes
Li is High
No
Li Off is Received
Yes
Li is Low
No
No
Month is Done
Yes
Calculate Year Bill
Send SMS Owner
Reset Parameters
Bill Amount is 1500
Yes
Send SMS to ministry
Reset Year Parameters
No

ALGORITHM

Step 1: Initialize Data Structures

- Create a User structure containing user ID, name, and a list of bills.
- Create a Bill structure with bill ID, amount, and due date.
- Create a Queue structure to store Notification messages with standard operations: enqueue, dequeue, is Empty, and is Full.

Step 2: Add Users and Bills

- Input user details.
- For each user, input multiple bills along with their due dates.
- Store this data in appropriate arrays or lists.

Step 3: Check for Due Bills

- For each bill of each user:
    - Get the current date.
    - Calculate the difference between the due date and current date.
    - If the bill is due within the next 7 days, proceed to the next step.

Step 4: Enqueue Notifications

- Create a message: "Reminder for [User]: Bill #[ID] of $[amount] is due on [due date]".
- Use the enqueue function to add this message to the notification queue.

Step 5: Display or Send Notifications

- While the queue is not empty:
    - Use the dequeue function to retrieve the earliest notification.
    - Display or simulate sending the notification to the user.

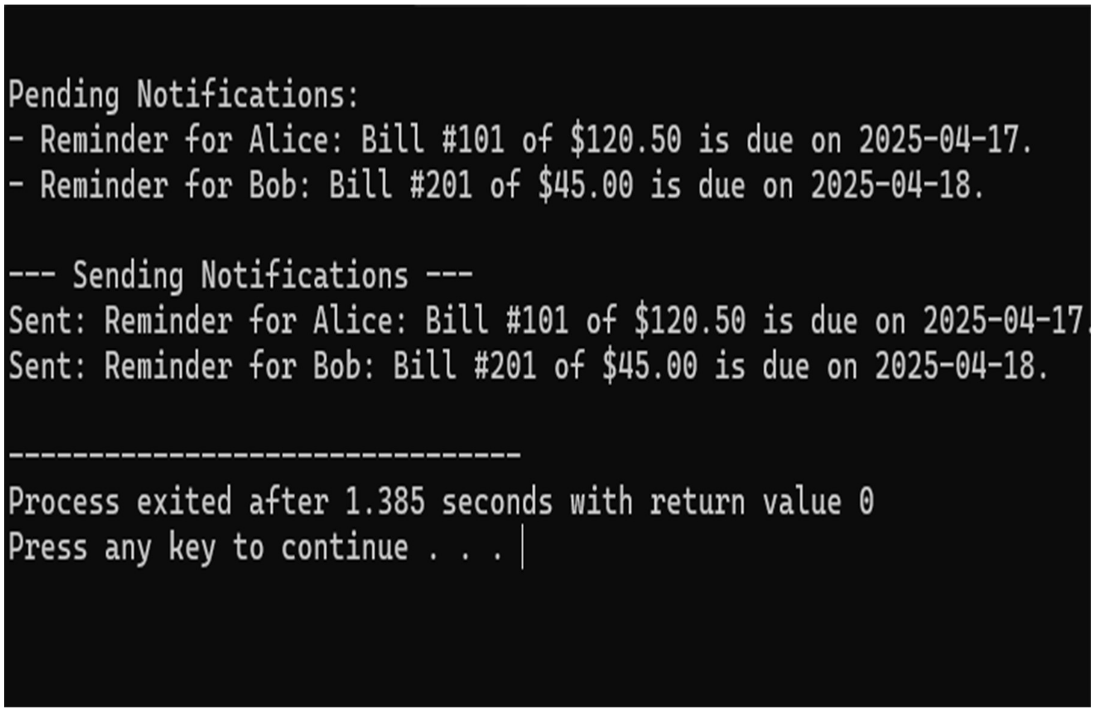
Step 6: End

- After all notifications are processed, terminate the program or loop.

IMPLEMENTATION

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

#define MAX_BILLS 10
#define MAX_USERS 5
#define MAX_NOTIFICATIONS 100


typedef struct {
   int bill_id;
   float amount;
   struct tm due_date;
} Bill;

typedef struct {
   int user_id;
   char name [50];
   Bill bills [MAX_BILLS];
   int bill_count;
} User;

typedef struct {
   char message [200];
} Notification;

typedef struct {
   Notification notifications [MAX_NOTIFICATIONS];
   int front, rear;
} NotificationQueue;


void initQueue(NotificationQueue *q) {
    q->front = q->rear = -1;
}

int isFull(NotificationQueue *q) {
    return q->rear == MAX_NOTIFICATIONS - 1;
}

int isEmpty(NotificationQueue *q) {
    return q->front == -1 || q->front > q->rear;
}

void enqueue(NotificationQueue *q, char *message) {
    if (isFull(q)) {
        printf("Notification queue is full!\n");
        return;
    }
    if (q->front == -1) q->front = 0;

    q->rear++;
    strncpy(q->notifications[q->rear].message, message,
sizeof(q>notifications[q]>rear].message));
}

void dequeue(NotificationQueue *q) {
    if (isEmpty(q)) {
        printf("No notifications to dequeue.\n");
        return;
    }
    printf("Sent: %s\n", q->notifications[q->front].message);
    q->front++;
}

void displayQueue(NotificationQueue *q) {
    if (isEmpty(q)) {
        printf("No pending notifications.\n");
        return;
    }
    printf("\nPending Notifications:\n");
    for (int i = q->front; i <= q->rear; i++) {
        printf("- %s\n", q->notifications[i].message);
    }
}

int is_due_soon(struct tm due_date) {
    time_t now = time(NULL);
    time_t bill_time = mktime(&due_date);

    double diff_days = difftime(bill_time, now) / (60 * 60 * 24);
    return (diff_days >= 0 && diff_days <= 7);
}

void add_bill(User *user, int bill_id, float amount, int year, int month, int day) {
    if (user->bill_count >= MAX_BILLS) return;

    Bill *b = &user->bills[user->bill_count++];
    b->bill_id = bill_id;
    b->amount = amount;
    b->due_date.tm_year = year - 1900;
    b->due_date.tm_mon = month - 1;
    b->due_date.tm_mday = day;
    b->due_date.tm_hour = 0;
    b->due_date.tm_min = 0;
    b->due_date.tm_sec = 0;
}

// Check bills and enqueue notifications
void check_bills_and_notify(User users[], int user_count, NotificationQueue *q) {
    char buffer[200];

    for (int i = 0; i < user_count; i++) {
        for (int j = 0; j < users[i].bill_count; j++) {
            Bill *b = &users[i].bills[j];
            if (is_due_soon(b->due_date)) {
                snprintf(buffer, sizeof(buffer),
                         "Reminder for %s: Bill #%d of $%.2f is due on %d-%02d-%02d.",
                         users[i].name, b->bill_id, b->amount,
                         b->due_date.tm_year + 1900,
                         b->due_date.tm_mon + 1,
                         b->due_date.tm_mday);
                enqueue(q, buffer);
            }
        }
    }
}


int main() {
    User users[MAX_USERS];
    int user_count = 0;
    NotificationQueue q;
    initQueue(&q);

    // Add users
    strcpy(users[user_count].name, "Alice");
    users[user_count].user_id = 1;
    users[user_count].bill_count = 0;
    user_count++;

    strcpy(users[user_count].name, "Bob");
    users[user_count].user_id = 2;
    users[user_count].bill_count = 0;
    user_count++;

    // Add bills (update dates to be within 7 days for testing)
    add_bill(&users[0], 101, 120.50, 2025, 4, 17);  // Due soon
    add_bill(&users[0], 102, 80.25, 2025, 4, 25);   // Not soon
    add_bill(&users[1], 201, 45.00, 2025, 4, 18);  // Due soon

    // Enqueue due bills
    check_bills_and_notify(users, user_count, &q);

    // Display all queued notifications
    displayQueue(&q);

    // Simulate sending notifications
    printf("\n--- Sending Notifications ---\n");
    while (!isEmpty(&q)) {
        dequeue(&q);
    }

    return 0;
}
```

OUTPUTS



```
Pending Notifications:
- Reminder for Alice: Bill #101 of $120.50 is due on 2025-04-17.
- Reminder for Bob: Bill #201 of $45.00 is due on 2025-04-18.

--- Sending Notifications ---
Sent: Reminder for Alice: Bill #101 of $120.50 is due on 2025-04-17
Sent: Reminder for Bob: Bill #201 of $45.00 is due on 2025-04-18.

--------------------------------
Process exited after 1.385 seconds with return value 0
Press any key to continue . . .
```

CONCLUSION

In conclusion, the bill notification system implemented using a queue in C demonstrates an efficient and structured approach to managing timely reminders for due payments. By utilizing the queue data structure, the system ensures that notifications are handled in a first-in-first-out (FIFO) manner, meaning bills that are due sooner are prioritized and processed first. This mirrors real-world scenarios where users should be alerted in the order of urgency. The program checks each user's bills against the current date and enqueues notifications for those that are due within the next seven days. Once enqueued, notifications can be displayed and processed sequentially, simulating how a real billing system might send alerts via SMS or email. This method not only organizes notifications effectively but also provides a scalable foundation for more complex features like priority handling, database integration, or user interaction. Overall, it showcases how fundamental data structures like queues can be applied to practical, real-life applications in billing and notification systems.