Library Management System – Full Code Documentation (Formatted for Submission + Viva)

This document is a clean, readable, professional-quality explanation of your **minimalist Library Management System in C**, rewritten and organized so you can submit it as project documentation and confidently survive any viva exam.

It contains: - System overview - Architecture diagrams (ASCII) - Code snippets (formatted) - Function-by-function explanation - File structure snapshots - Usage walkthrough - Viva questions + how to answer

---

1. System Overview

A lightweight **C console application** for handling a small library. It uses **two text files** (books.txt and members.txt) with **pipe-delimited records** and supports basic circulation.

✔️Core Features

- Add, list, search, delete books
- Add and list members
- Issue one book per member
- Return issued books
- Fully persistent across restarts

✔️Why this design?

- Extremely simple to understand
- Perfect for viva explanation
- No dynamic memory → no memory leaks
- No database → no extra installation required
- Plain text files make debugging easy

---

2. Configuration & Data Structures

Constants

```
#define DB_BOOK   "books.txt"
#define DB_MEMBER "members.txt"
#define MAX       1024      /* Buffer size for fgets */
#define MAXB      256       /* Max books in array */
```

```c
#define MAXM      256       /* Max members */
#define FLEN      60        /* Max length of title / author / name */
```

Purpose

- Centralizes all "limits" in one place
- Easy to change DB size or field length later
- Avoids magic numbers

---

3. Data Structures

Book Structure

```c
typedef struct {
    char title[FLEN];
    char author[FLEN];
    int  year;
} Book;
```

Explanation

- `title` – up to 59 characters
- `author` – up to 59 characters
- `year` – publication year

Used to store **book metadata**.

Member Structure

```c
typedef struct {
    int  id;
    char name[FLEN];
    char current[FLEN];  /* issued book, empty if none */
} Member;
```

Explanation

- `id` – numeric member ID
- `name` – member's name
- `current` – book they have issued (empty = none)

This simplifies circulation: each member can issue **at most one** book.

---

4. File I/O Operations

The system loads data from files at startup and rewrites files after every change.

4.1 Loading Books

```
int load_books(Book *b) {
    FILE *f = fopen(DB_BOOK, "r");
    int n = 0;
    if (!f) return 0;

    while (n < MAXB &&
           fscanf(f, " %60[^|]|%60[^|]|%d\n", b[n].title, b[n].author,
&b[n].year) == 3)
        ++n;

    fclose(f);
    return n;
}
```

Purpose

- Reads all books from **books.txt**
- Uses `fscanf` with `[^|]` to safely read fields without overflow
- Returns number of books loaded

Viva Point

- Missing file = first run = returns empty DB.

4.2 Saving Books

```
void save_books(Book *b, int n) {
    FILE *f = fopen(DB_BOOK, "w");
    for (int i = 0; i < n; i++)
        fprintf(f, "%s|%s|%d\n", b[i].title, b[i].author, b[i].year);
    fclose(f);
}
```

Purpose

- Writes full array back to file
- Overwrites file each time (simple + safe)

4.3 Loading Members

```
int load_members(Member *m) {
    FILE *f = fopen(DB_MEMBER, "r");
    int n = 0;
    if (!f) return 0;

    while (n < MAXM &&
           fscanf(f, " %d|%60[^|]|%60[^\n]\n",
&m[n].id, m[n].name, m[n].current) == 3)
        ++n;

    fclose(f);
    return n;
}
```

4.4 Saving Members

```
void save_members(Member *m, int n) {
    FILE *f = fopen(DB_MEMBER, "w");
    for (int i = 0; i < n; i++)
        fprintf(f, "%d|%s|%s\n", m[i].id, m[i].name,
m[i].current);
    fclose(f);
}
```

5. String Utility: strip()

```
void strip(char *s) {
    char *p = s, *q = s + strlen(s) - 1;
    while (isspace(*p)) p++;
    while (q > p && isspace(*q)) q--;
```

```
    q[1] = 0;
    memmove(s, p, strlen(p) + 1);
}
```

Purpose

- Removes leading and trailing spaces
- Cleans input from `fgets`

---

6. Member Management

6.1 Add Member

```c
void add_member(Member *m, int *nm) {
    if (*nm >= MAXM) { puts("Member DB full."); return; }

    printf("ID   : "); scanf("%d", &m[*nm].id); getchar();
    printf("Name : "); fgets(m[*nm].name, FLEN, stdin); strip(m[*nm].name);

    m[*nm].current[0] = 0;
    (*nm)++;
    save_members(m, *nm);

    puts("Member added.");
}
```

Logic

- Read ID and name
- Initialize `current = ""`
- Save updated list to file

6.2 List Members

```
void list_members(Member *m, int nm) {
    if (!nm) { puts("No members."); return; }

    puts("\nID  Name                           Issued
book");
    for (int i = 0; i < nm; i++)
        printf("%-3d %-28s %s\n",
               m[i].id, m[i].name,
               m[i].current[0] ? m[i].current : "-
none-");
}
```

7. Helper Search Functions

Find Member by ID

```
int find_member(Member *m, int nm, int id) {
    for (int i = 0; i < nm; i++) if (m[i].id == id) return i;
    return -1;
}
```

Find Book by Title

```
int find_book(Book *b, int nb, const char *title) {
    for (int i = 0; i < nb; i++) if
(strcmp(b[i].title, title) == 0) return i;
    return -1;
}
```

8. Circulation (Issue / Return)

8.1 Issue Book

```c
void issue_book(Book *b, int nb, Member *m, int nm) {
    int id; char title[FLEN];

    printf("Member ID : "); scanf("%d", &id);
getchar();
    int mi = find_member(m, nm, id);
    if (mi < 0) { puts("ID not found."); return; }
    if (m[mi].current[0]) { puts("Already has a
book."); return; }

    printf("Book title: "); fgets(title, FLEN,
stdin); strip(title);
    int bi = find_book(b, nb, title);
    if (bi < 0) { puts("Book not found."); return; }

    for (int i = 0; i < nm; i++)
        if (strcmp(m[i].current, title) == 0) {
            printf("Book already issued to %s\n",
m[i].name);
            return;
        }

    strcpy(m[mi].current, title);
    save_members(m, nm);

    puts("Book issued.");
}
```

8.2 Return Book

```c
void return_book(Member *m, int nm) {
    int id;
    printf("Member ID returning: "); scanf("%d",
&id); getchar();

    int mi = find_member(m, nm, id);
    if (mi < 0) { puts("ID not found."); return; }
    if (!m[mi].current[0]) { puts("No book to
return."); return; }

    printf("Returned: %s\n", m[mi].current);
    m[mi].current[0] = 0;
    save_members(m, nm);
}
```

9. Book Management

Add Book

```c
void add_book(Book *b, int *nb) {
    if (*nb >= MAXB) { puts("Book DB full."); return; }

    printf("Title  : "); fgets(b[*nb].title, FLEN, stdin); strip(b[*nb].title);
    printf("Author : "); fgets(b[*nb].author, FLEN, stdin);
strip(b[*nb].author);
    printf("Year   : "); scanf("%d", &b[*nb].year); getchar();

    (*nb)++;
    save_books(b, *nb);
    puts("Book added.");
}
```

List Books

```c void list_books(Book *b, int nb) { if (!nb) { puts("No books."); return; }
```

```
printf("\n%-35s %-25s %s\n", "TITLE", "AUTHOR", "YEAR");
for (int i = 0; i < nb; i++)
    printf("%-35s %-25s %d\n", b[i].title, b[i].author, b[i].year);
```