Previous year question papers management system

REQUIRMENTS:

SOFTWARE REQUIREMENTS:

- operating system : power point
- programming language : c programming
- compiler : c

HARDWARE REQUIREMENTS :

- 2GB RAM
- 512 MB Storage
- dual-core processor

INTRODUCTION:

The **Previous Year Question Paper Management System** is a smart digital platform designed to organize, store, and access question papers from past examinations in a fast and efficient way. In many educational institutions, students often struggle to find previous year papers due to improper storage, missing files, or unorganized records. This system solves that problem by providing a **centralized digital repository** where all question papers can be uploaded, categorized, and retrieved instantly.

Our system allows users to easily search for question papers based on **date, subject, and semester**, making the entire process smooth and user-friendly. Instead of wasting time searching manually through physical files or folders, students and faculty can access the needed papers with just a few clicks.

ALGORITHM:

STEP 1: start

STEP 2 : Display main menu

- Admin login
- Student login
- Register new student
- Exit

STEP 3 : INPUT user choice

STEP 4 : IF CHOICE = 1(Admin login)

- Input admin username and password
- VALIDATE credentials
- IF valid THEN
-

```
     Display admin menu
ELSE
     Display "Invalid credentials"
      GOTO step 2
STEP 5 : Admin menu
    1.   Upload question paper
    2.  Delete question paper
    3.  Update question paper
    4. View all question paper
    5. Manage categories/ subjects
    6.  View download statistics
    7.  Logout
    INPUT admin choice
STEP 6 : IF admin choice = (Upload Question Paper)
               . INPUT paper details
                 1. Subject name
                 2. Year
                 3. Semester
                 4. Exam type (mid/Final/Model)
                 5. Department
                 6. Regulation
    INPUT file to upload (pdf)
    VALIDATE file format
    GENERATE unique paper ID
    STORE paper details in database
    SAVE file to server
    Display "Upload Successful"
    GOTO Step 5
Step 7: ELSE IF admin choice = 2(Delete Quester Paper)
    Display list of all question papers
    INPUT paper ID to delete
    CONFORM deletion
    Display "Deletion successful"
    GOTO Step 5
Step 8: ELSE IF admin choice = 3(Update Question Paper)
    Display list of all question paper
    INPUT paper ID to update
    Display current details
    INPUT new details to update
    UPDATE database record
    Display "Update Successful"
    GOTO Step 5
```

Step 9: ELSE IF admin choice = 4(View All Question Paper)
- RETRIEVE all paper from database
- Display in tabular format with file

Step 10: ELSE IF admin choice = 5 (Manage Categories)
- Display sub-menu:
  1. Add Subject
  2. Delete Subject
  3. Add Department
  4. Delete Department
  5. Perform selected operation
  6. GOTO Step 5

Step 11: ELSE IF admin choice = 6 (View Statistics)
- Display download count for each paper
- Display most downloaded papers
- Display subject-wise statistics
- GOTO Step 5

Step 12: ELSE IF admin choice = 7
- Logout

Step 14

1. Search Question Papers
2. Browse by Subject
3. Browse by Year
4. Browse by Department
5. View My Downloads
6. Download Question Paper
7. Logout

INPUT student choice

step 15: IF student choice = 1 (Search)

INPUT search creterai+

Step 15: IF student choice = 1 (Search) INPUT search criteria:

- Keyword
- Subject
- year
- Semester
- Department
- SEARCH database with filters
- Display matching results
- GOTO Step 14

Step 16: ELSE IF student choice = 2 (Browse by Subject)

- Display list of all subjects
- SELECT subject
- Display all papers for that subject
- GOTO Step 14

Step 17: ELSE IF student choice = 3 (Browse by Year)

- Display list of available years
- SELECT year
- Display all papers for that year
- GOTO Step 14

Step 18: ELSE IF student choice = 4 (Browse by Department)

- Display list of departments
- SELECT department
- Display all papers for that department
- GOTO Step 14

Step 19: ELSE IF student choice = 5 (View My Downloads)

- RETRIEVE download history for logged-in student
- Display list with download dates
- GOTOStep14.

Step 20: ELSE IF student choice = 6 (Download)

- Display available question papers
- INPUT paper ID to download
- RETRIEVE file from server
- INCREMENT download counter
- RECORD download in student history
- DOWNLOAD file
- Display "Download Successful“
- GOTO Step 14

Step 21: ELSE IF student choice = 7

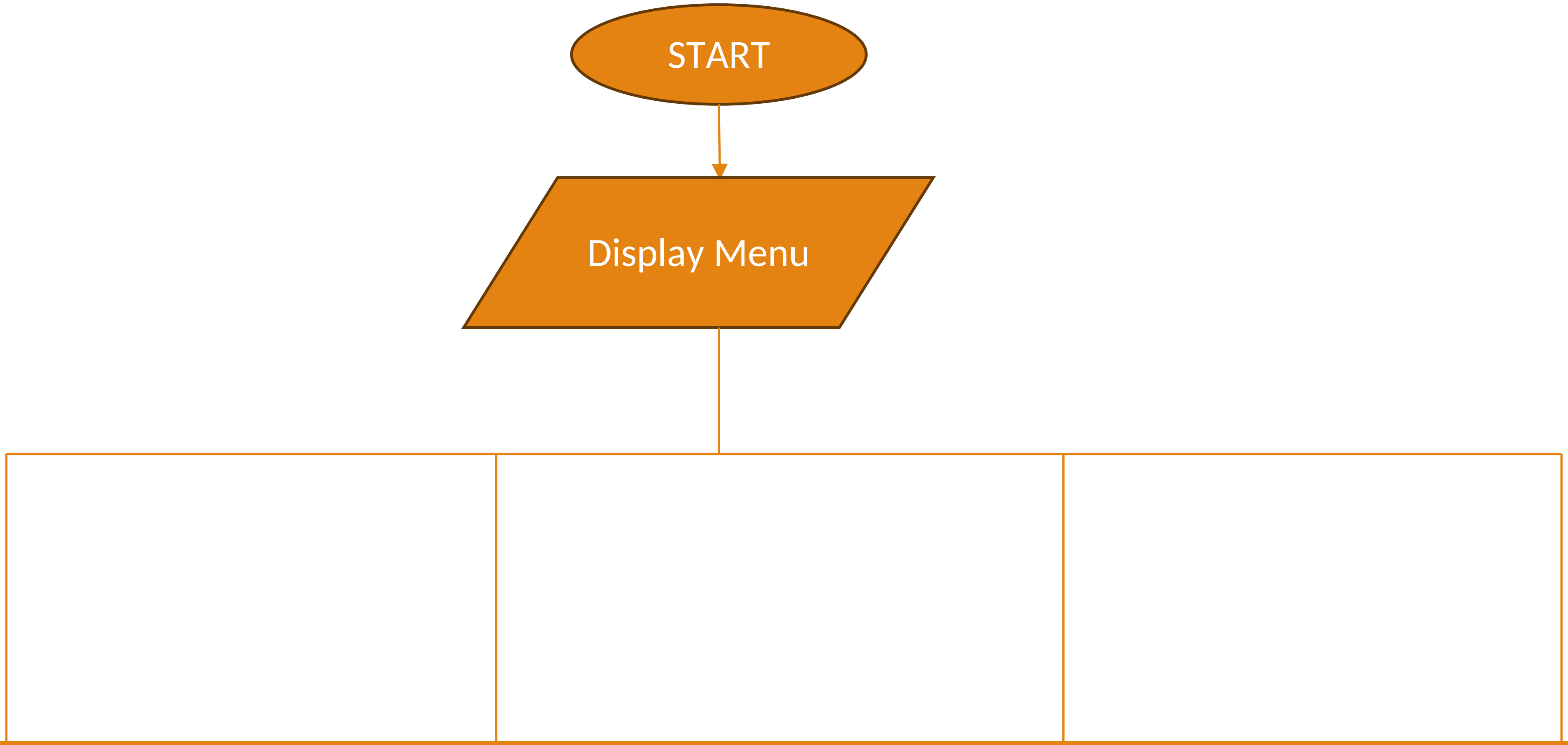
- Logout
- GOTO Step 2

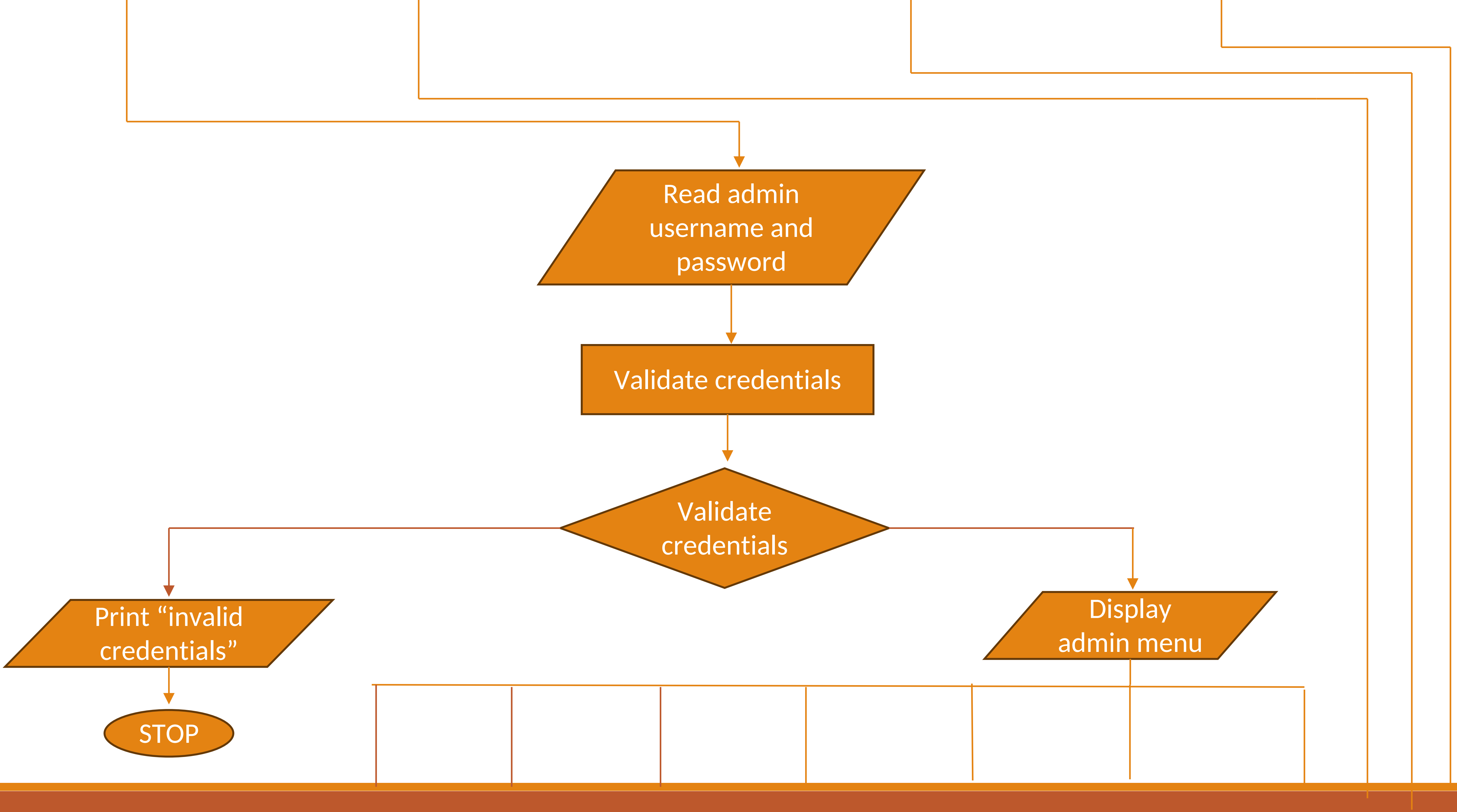
Step 22: IF choice = 3 (Register New Student)

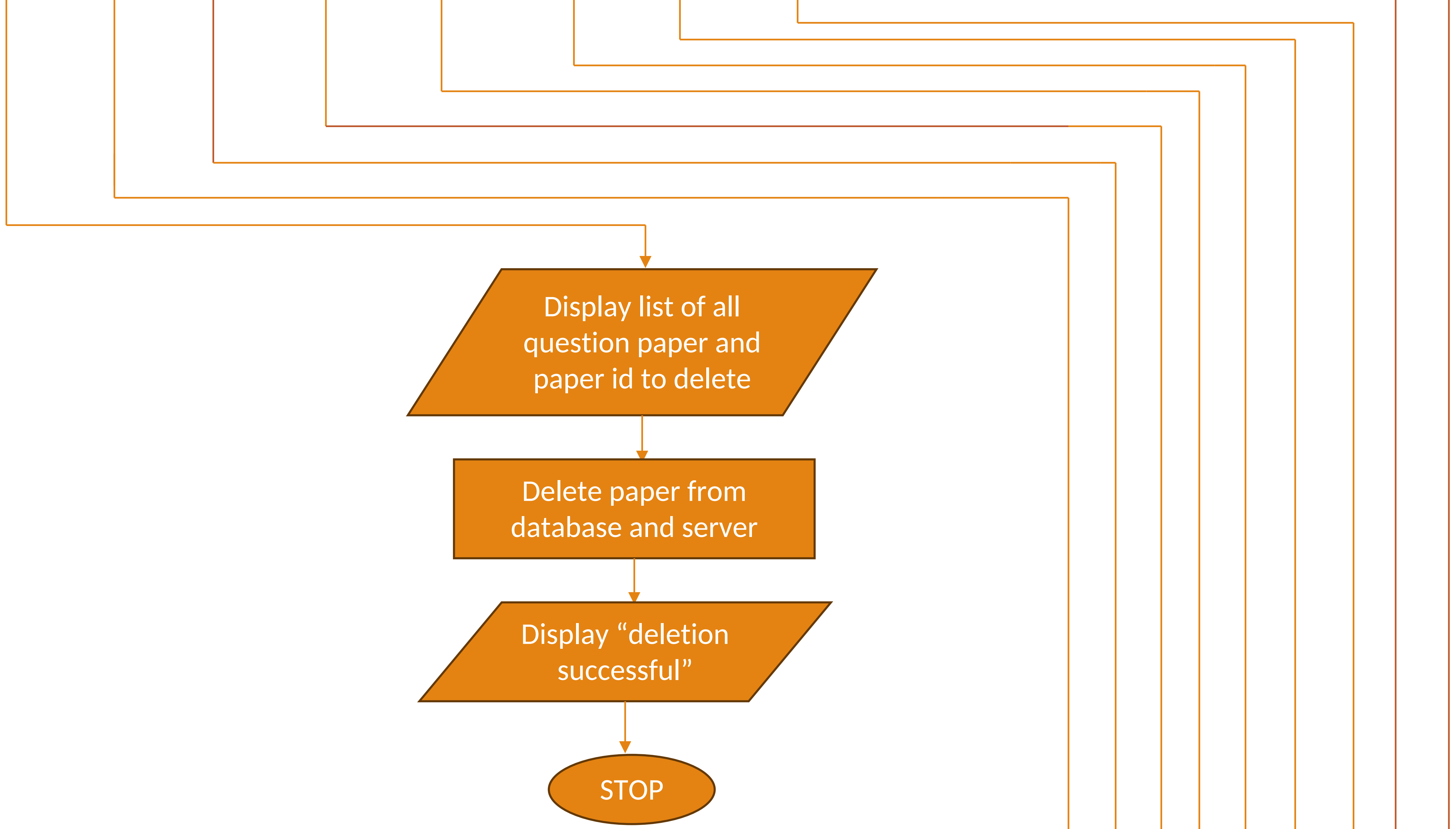
- INPUT student details:
  1. Name
  2. Roll Number
  3. Department
  4. Semester
  5. Email
  6. VALIDATE all fields

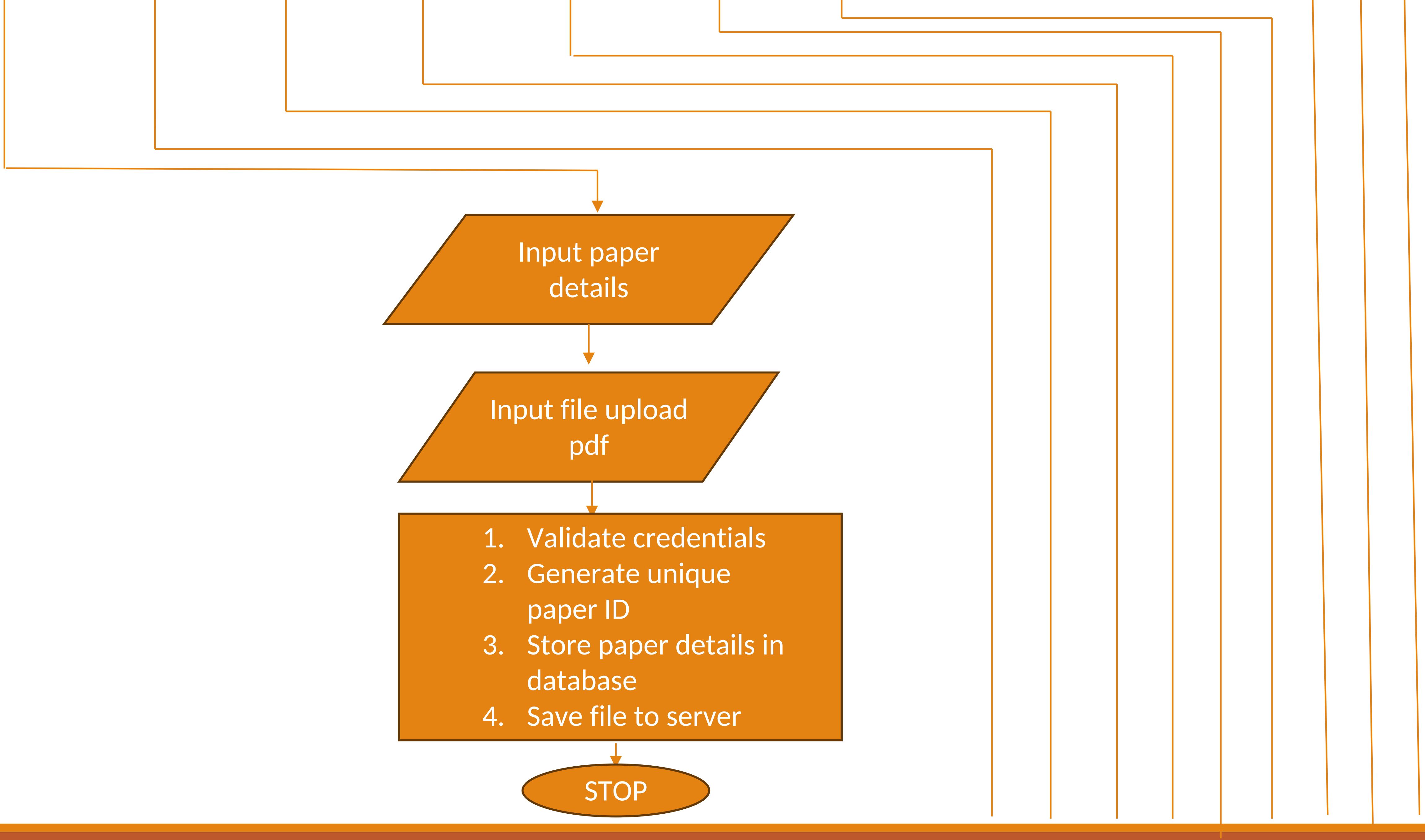
```
    7.  CHECK if roll number already exists
    8.  IF exists THEN
    9.  Display "Already registered"
    10. ELSE
    11. STORE student details in database
    12. Display "Registration successful"
    13. GOTO STEP 2
Step 23: IF choice = 4
    EXIT system
Step 24: STOP
```

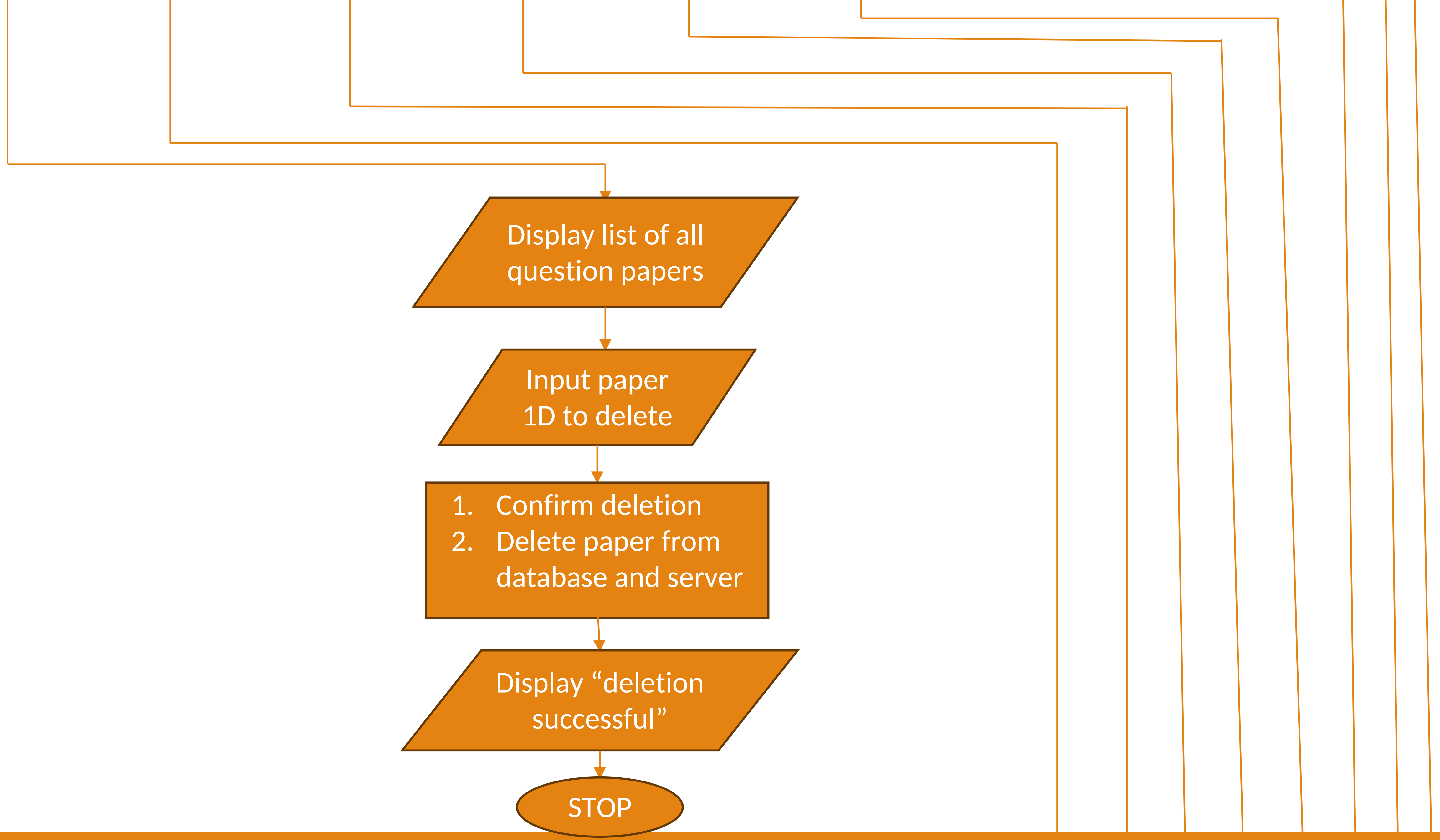
FLOWCHART :

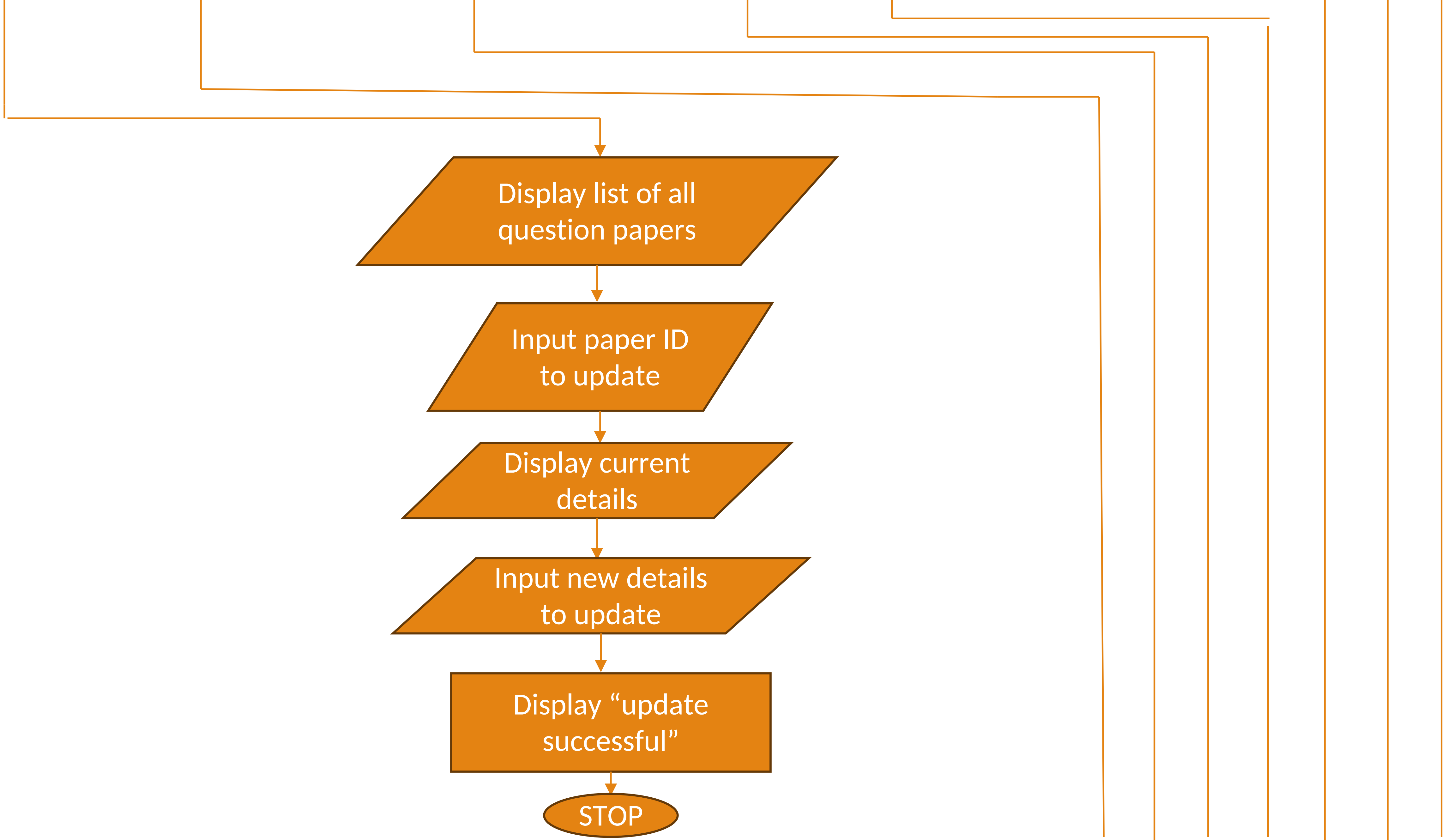


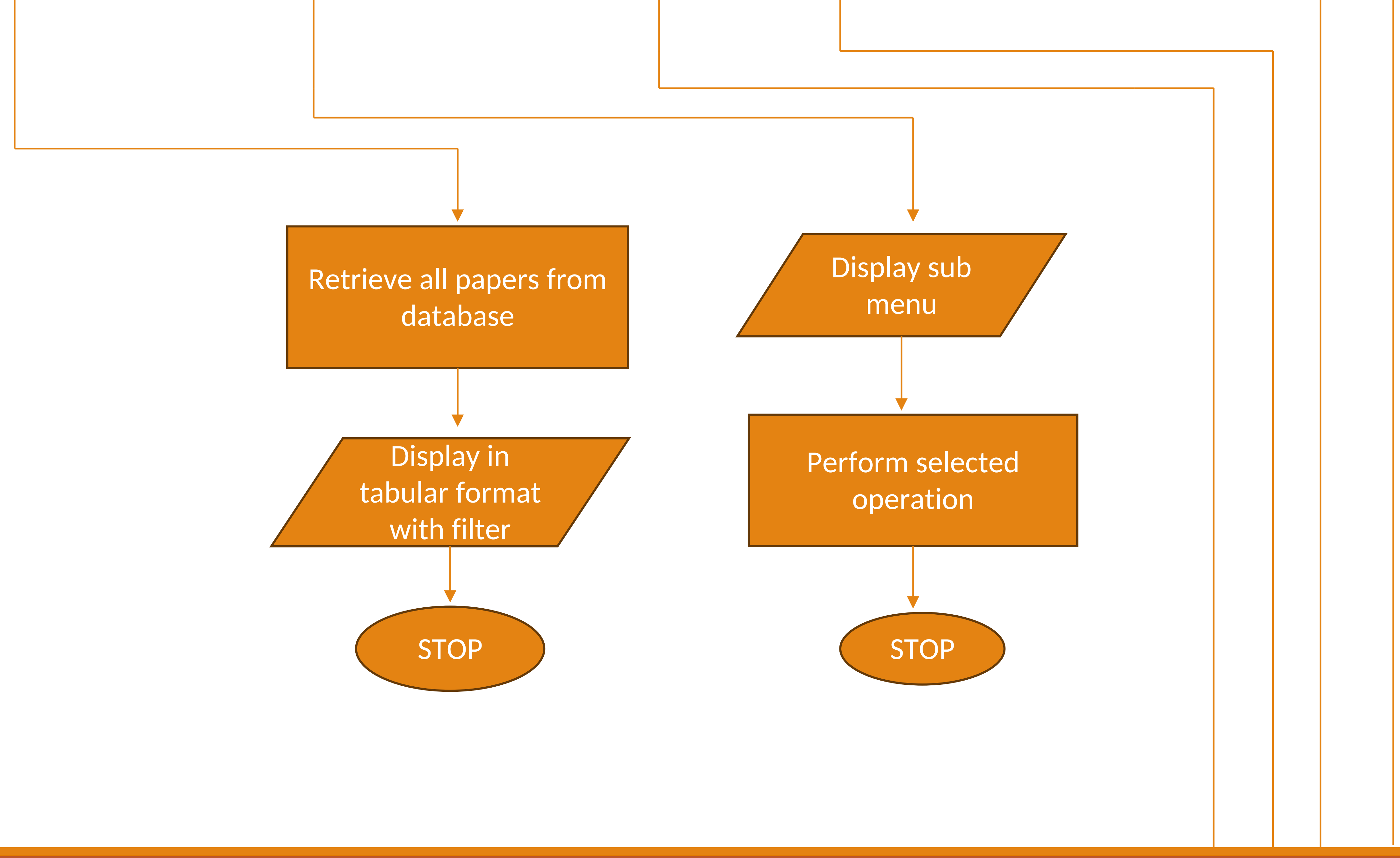

Read admin username and password
Validate credentials
Validate credentials
Print “invalid credentials”
STOP
Display admin menu

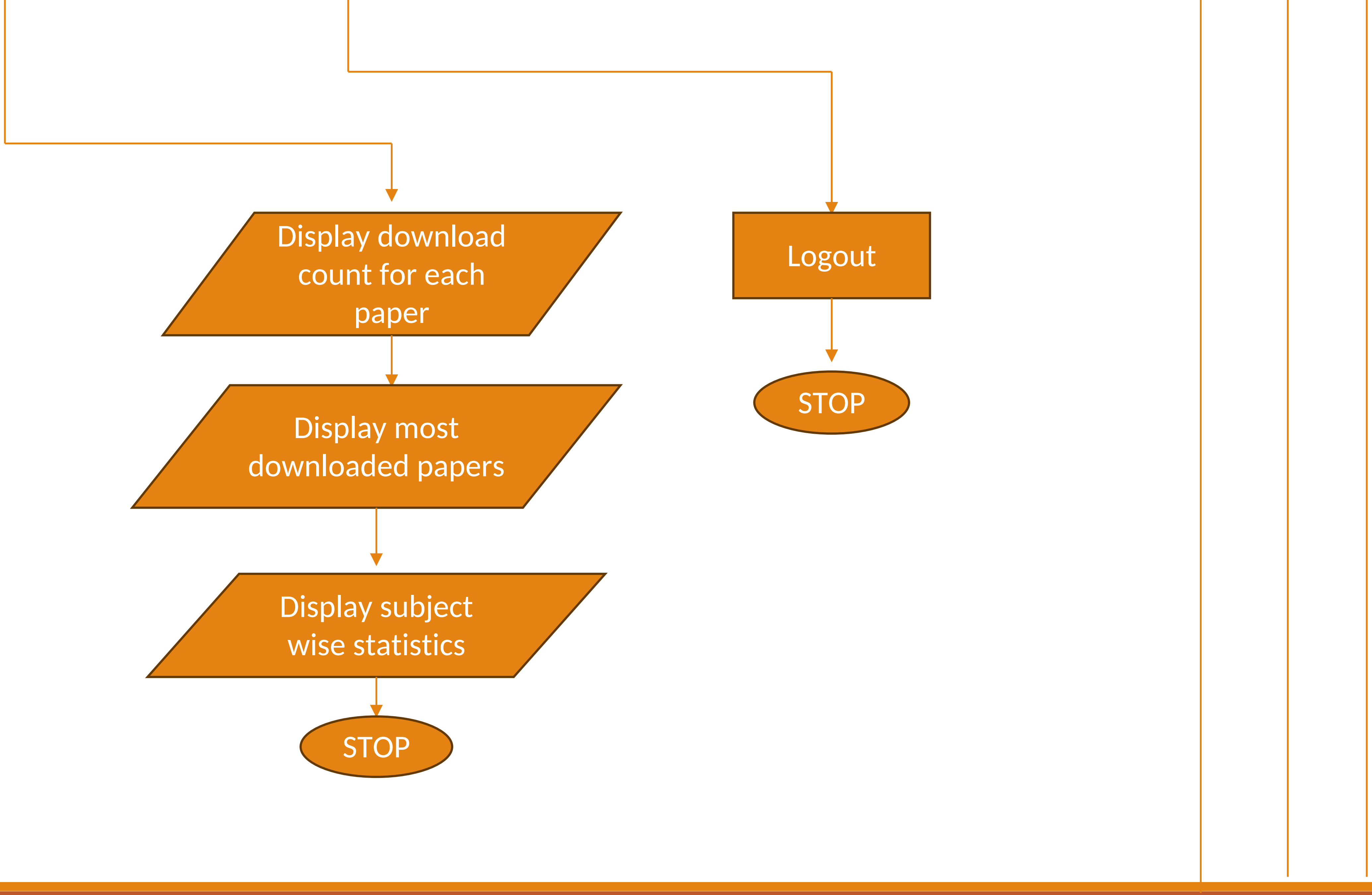

Display list of all question paper and paper id to delete
Delete paper from database and server
Display “deletion successful”
STOP

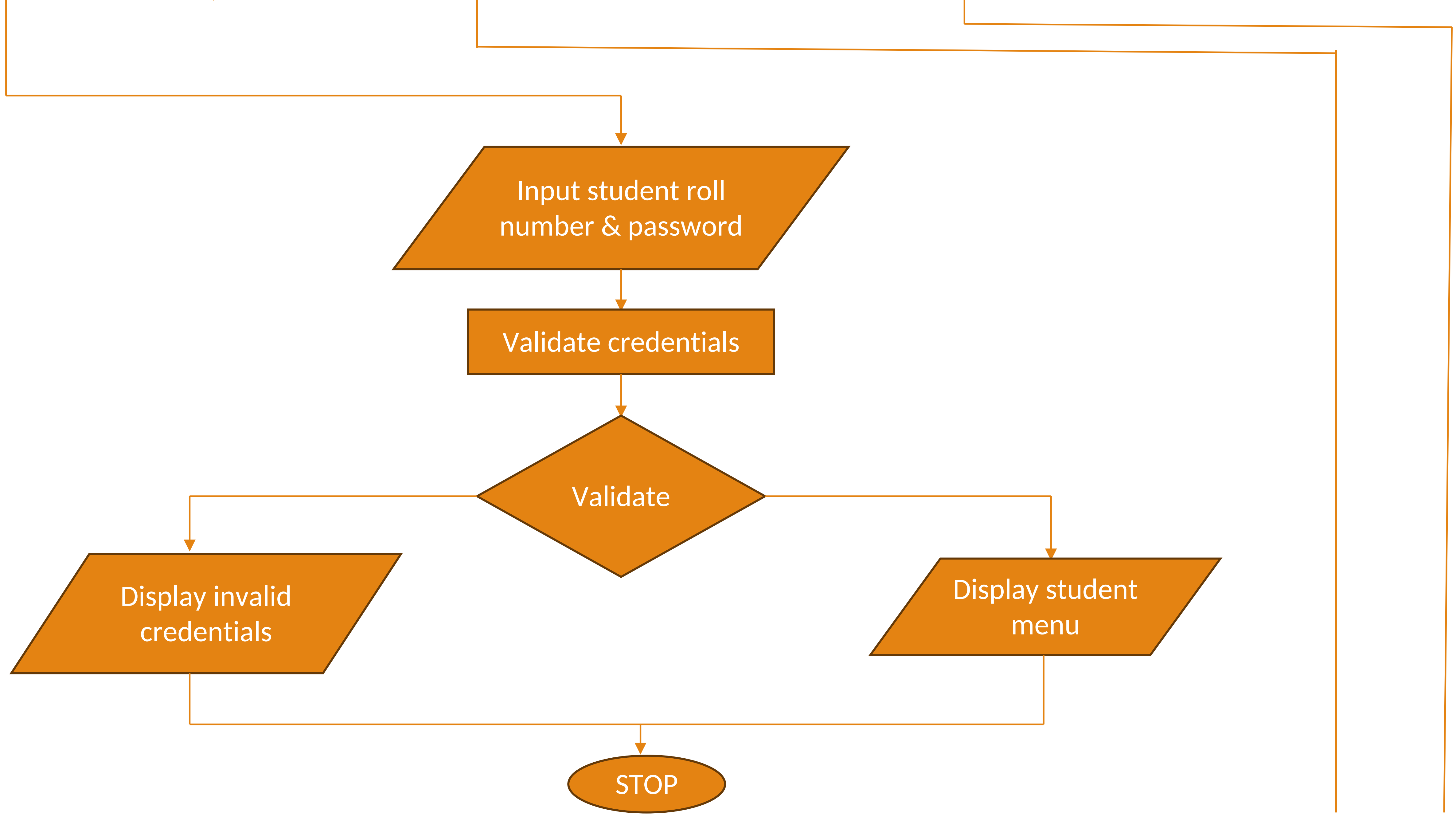

Input paper details
Input file upload pdf
1. Validate credentials
2. Generate unique paper ID
3. Store paper details in database
4. Save file to server
STOP

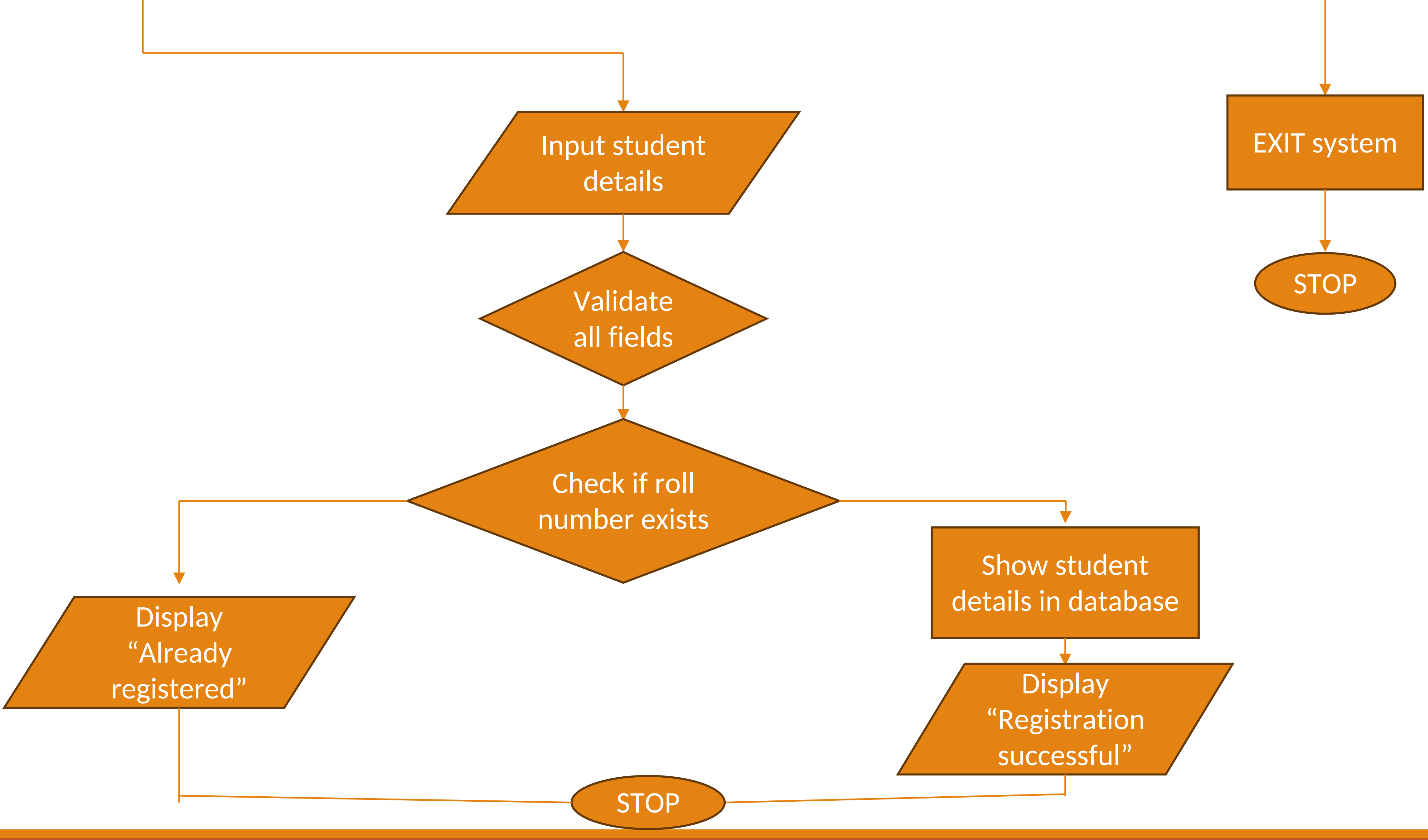

Display list of all question papers
Input paper 1D to delete
1. Confirm deletion
2. Delete paper from database and server
Display “deletion successful”
STOP


Display list of all question papers
Input paper ID to update
Display current details
Input new details to update
Display "update successful"
STOP


Retrieve all papers from database
Display in tabular format with filter
STOP
Display sub menu
Perform selected operation
STOP


Display download count for each paper
Display most downloaded papers
Display subject wise statistics
STOP
Logout
STOP


Input student roll number & password
Validate credentials
Validate
Display invalid credentials
Display student menu
STOP


Input student details
Validate all fields
Check if roll number exists
Display "Already registered"
Show student details in database
Display "Registration successful"
STOP
EXIT system
STOP

MODULE :

1. user Authentication/Registration Module

>Stores and manages user details (roll number, name, etc.),

>Allows login/verification before user download question papers.

>Ensures only valid users can access the system.

2. Question Paper Data Management Module

>stores the details of each question paper:

>paper ID

>Subject
>Year
>Semester
>Exam type
>Department
>Regulation
>Filename
>Download count
>Loads and saves question paper records from a file.

3. Add/Upload Question Paper Module

>Allows admin to enter a new paper’s details.
>Validates fields (tear,ID format, department, etc.).
>Stores the file name of the paper uploaded.

4.Search & Filter Module

>Subject
>Year
>Department
>Regulation
>exam type
>semester
provides fast access to required paper.

5.Display Question Paper List Module

>Shows all papers in a proper table format.
>Can show papers sorted by:
>year
>Subject alphabetically
>Department

6. Download Question Paper Module

>Manages downloading paper files.
>Increases the download count for each paper.
>Tracks how many time each paper is accessed.

7.Download Count Analytics Module

>Displays:
>Most downloaded paper
>Least downloaded paper
>Department-wise download statistics

8. File Handling Module

Handles saving and loading data into txt or data files:
>Store all question paper records.
>Store user information.
>Update download counts.

9. Error Handling & Validation Module

Manages:
>Invalid inputs
>Missing files
>Duplicate paper IDs
>Incomplete fields

10. Main Menu/User Interface Module

>Provides the user with:
>Login
>Add question paper
>Search paper
>Download
>View statistics
>Exit
>Connects all modules together

MODULE WISE LOGIC :

> If not, display an error message and deny access.
> Student Registration Logic
> Ask for student details: name, roll number, email, and password.
> Check whether the roll number already exists in the system.
> If it exists, display an error message.
> If it does not exist, save the details to the student database.
> Show a "Registration Successful" message.

2. Admin Operations Module – Logic

> Upload Question Paper Ask admin to enter subject, year, semester, exam type, department, regulation, and file name.
>Check whether the file has a valid PDF extension.
> Generate a unique paper ID
> Store all entered details in the paper database.
> Save the updated database to file.
> Display a success message.
> Delete Question Paper
> Display a list of all available question papers along with their IDs.
> Ask the admin to enter the Paper ID to delete.

> Search for the ID in the papers list.
> If found, remove that record by shifting the array elements.
> Save the updated data.
> Display a “Paper Deleted” confirmation message.
> Update Question Paper
> Show all question papers to the admin.
> Ask for the Paper ID that needs to be updated.
> Search for the given ID.
> If found, ask which field needs to be updated (subject, year, semester, etc.).
> Replace the old value with the new one.
> Save the updated data.
> Display an “Update Successful” message.
> View All Question Papers
> Loop through the paper database.
> Display each paper’s ID, subject, year, semester, department, and exam type.
> If the database is empty, show “No Papers Uploaded”.
> View Download Statistics
> Loop through the paper database.
> Display each paper’s ID, subject, and download count.

> Identify the paper with the highest downloads.
> Display the most downloaded paper details.

3. Student Operations Module – Logic

> Search Question Papers
> Ask the student to enter a keyword, subject, year, department, or semester.
> Loop through all papers and check for matches.
> If a match is found, display the paper details.
> If no matches exist, show “No Results Found”.
> Browse by Subject Extract all unique subjects from the paper list.
> Display the list of subjects.
> Ask the student to choose one subject.
> Display all papers belonging to that subject.
> Browse by Year List all available years from the database.
> Ask the student to select a year.
> Display all papers uploaded in that year.
> Browse by Department Show the list of departments.
> Let the student select a department.
> Display all papers related to that department.

>   Download Question Paper
>   Ask the student to enter the Paper ID to download.
> Search for the given ID.
> If found, increase the download count for that paper.
> Record the download in the student's download history along with the date.
> Save the updated data.
> Display a "Download Successful" message.
> View Download History Retrieve the download history of the logged-in student.
> If download records exist, display them with dates.
> If no history exists, show "No Downloads Yet".

4. **File Handling Module – Logic**

> Load Data()
> Open the student database file and load all student records.
> Open the papers database file and load all paper records.
> Open the download history file and load download data.
> Close all files after reading.
> Save Data()
> Open the data files for writing.
> Save all student records, paper records, and download history to the respective files.

Close the files.
Display a “Data Saved” confirmation message.

5. Utility Module – Logic

> Clear Input Buffer()
> Continuously read characters until a newline is detected.
> Helps prevent unwanted leftover input in the buffer.
> Generate Paper ID()
> Maintain a counter that increases with each new paper entry.
> Return the counter value as the new Paper ID.
> Get Current Date()
> Retrieve the current system date.
> Convert it into a readable format (e.g., DD-MM-YYYY).
> Return the date as a string.

MODULE PROGRAM :

1. Main Module

```
       1.1 main() – Logic Call load Data() to load existing data (question papers, students, downloads).
        Repeat forever:
        Call display Main Menu() to show main options.
        Read user choice.
Switch on choice:
     1 → call admin Login().
     2 → call student Login().
     3 → call register New Student().
     4 →Print exit message .
Call save Data().
Exit the program.
Default → print "Invalid choice" and continue loop.
 1.2 display Main Menu() – Logic
```

Print system title and decorative lines.
Print:
Admin Login
Student Login
Register New Student
Exit

2. Admin Module

2.1 admin Login() – Logic
Prompt admin to enter username.
Prompt admin to enter password.
Remove newline characters from both strings.
If username is "admin" and password is "admin123":
Print “Login Successful”.
Call admin Menu(username).
Else:
Print “Invalid Credentials”.
Return to main menu.
2.2 admin Menu(char *admin Username) – Logic
Repeat until admin chooses logout:
Display admin menu with options:
Upload Question Paper

```
    Delete Question Paper
    Update Question Paper
    View All Question Papers
    Manage Categories/Subjects
    View Download Statistics
    Logout
   Read admin choice.
Switch on choice:
    1 → upload Question Paper()
    2 → delete Question Paper()
    3 → update Question Paper()
    4 → view All Question Papers()
    5 → manage Categories()
    6 → view Download Statistics()
    7 → print “Logging out…” and return.
    Default → print “Invalid choice”.
```

3. Question Paper Management Module

```
    3.1 upload Question Paper() – Logic
    Declare a new Question
    Prompt admin to enter:
     > Subject
```

```
  > Year
  > Semester
  > Exam type (Mid/Final/Model)
   > Department
  > Regulation
  > PDF filename
 Check if filename contains “.pdf”:
 If not, print “Invalid file format” and return.
 Call generate Paper ID(new Paper . paper ID) to generate unique ID.
 Set new Paper . download Count = 0.
 Store new Paper in papers[paper Count].
 Increment paper Count.
 Print upload success message and generated Paper ID.
 Call save Data().
```

3.2 delete Question Paper() – Logic

```
 If paper Count == 0:
 Print “No question papers available”.
 Return.
 Call view All Question Papers() to show list.
 Prompt admin to enter paper ID to delete.
 Search in papers[]:
 If found, store index in found . If not found, print “Paper ID not found” and return .
```

```
        Ask admin for confirmation (y/n).
        If confirmed:
        Shift elements from found + 1 to end one position left.
        Decrement paper Count.
        Print “Deletion successful”.
        all save Data().
        Else:
        Print “Deletion cancelled”
```

3.3 update Question Paper() – Logic

```
    If paper Count == 0:
    Print “No question papers available”.
    Return.
    Call view All Question Papers().
   Prompt admin to enter paper ID to update.
   Search in papers[]:
   If not found, print “Paper ID not found” and return.
   Display current details of that paper.
   Prompt admin to enter new values:
   New subject name
   New year
   New semester
   Update these fields in the selected papers[found].
```

```
      Print “Update successful
     Call save Data().
3.4 view All Question Papers() – Logic
     Print heading “All Question Papers”.
     If paper Count == 0:
     Print “No question papers available”.
     Return.
     Print table header (Paper ID, Subject, Year, Semester, Department).
     For i = 0 to paper Count - 1:
    Print details of papers[i].
3.5 manage Categories() – Logic
     Display menu:
     Add Subject
     Delete Subject
     Add Department
     Delete Department
     Back
     Read choice.
Switch on choice:
   1 → print “Subject management feature (to be implemented)”.
   2 → print “Subject deletion feature (to be implemented)”.
   3 → print “Department management feature (to be implemented)”.
```

```
    3 → print “Department management feature (to be implemented)”.
    4 → print “Department deletion feature (to be implemented)”.
    5 → return . Default → print “Invalid choice”.
```

3.6 view Download Statistics() – Logic

```
    Print heading “Download Statistics”.
    If paper Count == 0:
    Print “No question papers available”.
    Return.
    Print table header (Paper ID, Subject, Downloads).
    For each paper:
    Print Paper ID, Subject, and download Count.
    Initialize max Downloads = 0, max Index = 0.
    For each paper:
    If papers[i].download Count > max Downloads:
    Update max Downloads and max Index.
    If max Downloads > 0:
    Print the most downloaded paper subject and count.
```

4. Student Module

4.1 student Login() – Logic

```
   Prompt user to enter roll number.
   Prompt user to enter password.
   Initialize found = -1.
```

```
Loop from i = 0 to student Count - 1:
If students[i].roll Number matches and students[i].password matches:
Set found = i and break.
If found != -1:
Print “Login successful” and greet with student name.
Call student Menu(roll Number).
Else:
Print “Invalid credentials”.
```

4.2 register New Student() – Typical Logic

```
Prompt for student details : Roll number, name, department, semester, email, password.
Check that roll number is not already present.
Create a Student structure with entered data.
Store in students[student Count].
Increment student Count.
Call save Data().
```

4.3 student Menu(char *roll Number) – Logic

```
Repeat until student logs out:
Display Student Menu:
Search Question Papers
Browse by Subject
Browse by Year
Browse by Department
```

```
    View My Downloads
    Download Question Paper
    Logout
    Read student choice.
Switch on choice:
    1 → search Question Papers (roll Number)
    2 → browse By Subject(roll Number)
    3 → browse By Year(roll Number)
    4 → browse By Department(roll Number)
    5 → view My Downloads(roll Number)
    6 → download Question Paper(roll Number)
    7 → print “Logging out…” and return.
    Default → print “Invalid choice”.
```

5. Search and Browse Module

```
5.1 search Question Papers(char *roll Number) – Logic
    Prompt for search criteria:
    Keyword (can be part of subject, optional)
    Subject (exact, optional)
    Year (0 to skip)
    Semester (optional)
    Department (optional)
    Initialize found = 0.
```

```
For each paper:
Assume match = 1.
If keyword is not empty and subject does not contain keyword → match = 0.
If subject is not empty and unequal → match = 0.
If year > 0 and not equal → match = 0.
If semester is not empty and unequal → match = 0.
If department is not empty and unequal → match = 0.
If match == 1:
Print details of that paper.
    Increment found.
     After loop:
If found == 0: print “No matching results found”.
Else: print total number of results.
5.2 browse By Subject(char *roll Number) – Logic
     If paper Count == 0:
     Print “No question papers available”.
     Return.
    Build a list of unique subjects:
    For each paper, check if its subject already exists in subjects[].
    If not, add it and increment subject Count. Display all subjects with numbers (1 to subjectCount).
    Prompt user to select subject number.
    If invalid number : Print “Invalid choice” and return.
```

Else, for each paper:
If paper’s subject matches selected subject:
Print Paper ID, Year, Semester, etc.

5.3 browse By Year(char *roll Number) – Logic

If paper Count == 0:
Print “No question papers available”.
Return.
Build a list of unique years:
For each paper, if its year is not already in years[], add it.
Display all years with numbers.
Prompt user to select year number.
If invalid: Print “Invalid choice” and return.
Else, for each paper:
If paper’s year equals selected year:
Print Paper ID, Subject, Semester, etc.

5.4 browse By Department(char *roll Number) – Logic

If paper Count == 0:
Print “No question papers available”.
Return . Build a list of unique departments:
For each paper, if department not already in departments[], add it.

```
Display departments with numbers.
Prompt user to select department number.
If invalid:
Print “Invalid choice” and return.
Else, for each paper:
If paper’s department matches selected one:
Print Paper ID, Subject, Year, Semester.
```

6. Download and History Module

6.1 download Question Paper(char *roll Number) – Logic

```
Prompt student to enter Paper ID to download (optionally after showing/searching list).
Search for this Paper ID in papers[]:
If not found, print “Paper ID not found” and return.
If found : Increment papers[index].download Count.
Create a new Download History record:
Roll Number = given roll number.
Paper ID   = selected paper ID.
Download Date = current date string from get Current Date().
Store it in downloads[download Count]. Increment download Count.
Print a message simulating successful download, with filename.
Call save Data().
```

6.2 view My Downloads(char *roll Number) – Logic

```
    Print heading “My Downloads”.
    Initialize found = 0.
    Loop through all entries in downloads[]:
    If downloads[i].roll Number equals given roll Number:
    Print Paper ID and Download Date.
    Optionally, look up subject from papers[].
    Increment found . If found == 0:
    Print “You have not downloaded any papers yet”.
```

7. Utility and Data Handling Module

```
7.1 clear Input Buffer() – Logic
    Loop : Read a character using get char().
    If character is newline '\n' or EOF, break loop.
    Used after scanf to remove leftover newline characters.
7.2 generate Paper ID(char *id) – Typical Logic
    Use current paper Count + 1 as numeric part.
    Format paper ID string, e.g. "P001", "P002", etc.
    Store formatted ID into id using sprint
7.3 get Current Date(char *date) – Typical Logic
    Get current system time using time() and local time().
```

Format it into a readable string using strftime,

e.g. "dd-mm-yyyy hh: mm: ss".

Store formatted string in date.

7.4 load Data() – Logic

Open data files (for papers, students, downloads) if they exist.

Read counts (paper Count, student Count, download Count).

Read corresponding arrays (papers[], students[], downloads[]).

If files do not exist, initialize counts to 0.Close all files.

7.5 save Data() – Logic

Open data files for writing.

Write paper Count and all papers[] to file.

Write student Count and all students[].

Write download Count and all downloads[].

Closeallfiles.

MODULE INTEGERATION :

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

// Structure Definitions type def struct {
char paper ID[20];
char subject[50];
int year;
char semester[20];
char exam Type[20];
char department[50];
char regulation[30];
char filename[100];
int download Count;
}
```

```
Question Paper;
typedef struct {
char roll Number[20];
char name[50];
char department[50];
char semester[20];
char email[50];
char password[20];
}
Student;
typedef struct {
char username[30];
char password[30];
}
Admin;
typedef struct {
char roll Number[20];
char paper ID[20];
char download Date[30];
}
```

```
Download History;

// Global Variables Question Paper papers[500];
Student students[200];
Download History downloads[1000];
int paper Count = 0;
int student Count = 0;
int download Count = 0;

// Function Proto types void display Main Menu();
void admin Login();
void student Login();
void register New Student();
void admin Menu(char *admin Username);
void student Menu(char *roll Number);
void upload Question Paper();
void delete Question Paper();
void update Question Paper();
void view All Question Papers();
void manage Categories();
void view Download Statistics();
void search Question Papers(char *roll Number);
void browse By Subject(char *roll Number);
void browse By Year(char *roll Number);
void browse By Department(char *roll Number);
void view My Downloads(char *roll Number);
void download Question Paper(char *roll Number);
void load Data();
void save Data();
void clear Input Buffer();
void generate Paper ID(char *id);
void get Current Date(char *date);

// Main Function
int main() {
int choice;
Load Data();
while(1) {
// Step 1: START
// Step 2: Display Main Menu
Display Main Menu();
```

```
// Step 3: INPUT user choice
Printf ("\n Enter your choice: ");
Scanf ("%d", &choice);
Clear Input Buffer();

// Step 4-23: Process choices
switch(choice) {
case 1:
Admin Login();
break;
case 2:
Student Login();
break;
case 3:
Register New Student();
break;
case 4:

// Step 23: EXIT system
Printf ("\n Thank you for using Question Papers Management System!\n");
Save Data();
exit(0);
default:
printf("\n Invalid choice! Please try again.\n"); }
}
return 0; }
void display Main Menu() {
Printf ("\n========================================\n");
Printf (" QUESTION PAPERS MANAGEMENT SYSTEM\n"); printf("========================================\n");
Printf ("1. Admin Login\n");
Printf ("2. Student Login\n");
Printf ("3. Register New Student\n");
Printf ("4. Exit\n");
Printf ("========================================\n");}
void admin Login() {
char username[30], password[30];
Printf ("\n--- ADMIN LOGIN ---\n");
Printf ("Enter username: ");
Fgets (username, size of(username), stdin);
```

```
password[strcspn(password, "\n")] = 0;

// Simple validation (admin/admin123)
if(strcmp(username, "admin") == 0 && strcmp(password, "admin123") == 0) {
printf("\n✓ Login Successful!\n");
Admin Menu(username); }
else {
printf("\n✗ Invalid Credentials!\n");
printf("Returning to main menu...\n"); } }
void admin Menu(char *admin Username) {
int choice;
while(1) {

// Step 5: Admin Menu
printf("\n========================================\n");
printf(" ADMIN MENU (Logged in as: %s)\n", adminUsername); printf("========================================\n");
printf("1. Upload Question Paper\n");
printf("2. Delete Question Paper\n");
printf("3. Update Question Paper\n");
printf("4. View All Question Papers\n");
printf("5. Manage Categories/Subjects\n");
printf("6. View Download Statistics\n");
printf("7. Logout\n");
printf("========================================\n");
printf("\n Enter your choice: ");
scanf("%d", &choice);
Clear Input Buffer();

switch(choice) {
case 1:
Upload Question Paper();
break;
case 2:
Delete Question Paper();
break;
case 3:
Update Question Paper();
break;
case 4:
View All Question Papers();
break;
case 5:
Manage Categories();
break;
case 6:
View Download Statistics();
break;
case 7:
printf("\n Logging out...\n");
return;
default:
printf("\n Invalid choice!\n");
}
}
}
void upload Question Paper() {
Question Paper new Paper;
printf("\n--- UPLOAD QUESTION PAPER ---\n");
printf("Enter Subject Name: ");
F gets(new Paper. subject, size of(new Paper . subject ), stdin);
New Paper. subject [strcspn(new Paper . subject, "\n")] = 0;

printf("Enter Year: ");
scanf("%d", &new Paper. year);
Clear Input Buffer();

printf("Enter Semester: ");
F gets(new Paper . semester, size of (new Paper. semester), stdin);
New Paper .semester [strcspn (new Paper. semester, "\n")] = 0;

printf("Enter Exam Type (Mid/Final/Model): ");
F gets(new Paper. Exam Type,
Size of(new Paper. Exam Type), stdin);
New Paper. Exam Type[strcspn(new Paper. exam Type, "\n")] = 0;

printf("Enter Department: ");
F gets(new Paper . department,
Size of (new Paper . department), stdin);
```

```
New Paper. department[strcspn( new Paper. department, "\n")] = 0;

printf("Enter Regulation: ");
F gets (new Paper. regulation,
Size of(new Paper. regulation), stdin);
New Paper. regulation[strcspn(new Paper. regulation, "\n")] = 0;

printf("Enter PDF filename: ");
F gets(new Paper. filename, size of(new Paper. filename), stdin); new Paper . filename[strcspn(new Paper. filename, "\n")] =
0;

// Validate file format (PDF)
if(strstr(new Paper . file name, ".pdf") == NULL) {
printf("\n✗ Invalid file format! Only PDF files allowed.\n");
return; }

// Generate unique paper ID
Generate Paper ID(new Paper. paper ID);
New Paper . Download Count = 0;
New Paper . department[strcspn(new Paper . department, "\n")] = 0;

printf("Enter Regulation: ");
fgets(new Paper. regulation,
Size of(new Paper . regulation), stdin);
New Paper . regulation[strcspn(new Paper . regulation, "\n")] = 0;

printf("Enter PDF filename: ");
F gets(new Paper . filename, size of(new Paper . filename), stdin); new Paper . filename[strcspn(new Paper . filename, "\n")]
= 0;

// Validate file format (PDF)
if(strstr(new Paper . filename, ".pdf") == NULL) {
printf("\n✗ Invalid file format! Only PDF files allowed.\n");
return; }

// Generate unique paper ID
Generate Paper ID(new Paper. Paper ID);
New Paper . download Count = 0;
```

```
// Find paper
for(int i = 0; i < paper Count; i++) {
if(strcmp(papers[i].paper ID, paper ID) == 0) {
found = i;
break; }
}
if(found == -1) {
printf("\n✗ Paper ID not found!\n");
return;
}

// Confirm deletion char confirm;
printf("\n Are you sure you want to delete this paper? (y/n): ");
scanf("%c", &confirm);
Clear Input Buffer();
if(confirm == 'y' || confirm == 'Y') {

// Delete paper from database
for(int i = found; i < paper Count - 1; i++) {
papers[i] = papers[i + 1]; }
Paper Count--;
printf("\n✓ Deletion Successful!\n");
Save Data(); }
else {
printf("\n Deletion cancelled.\n"); }
}
void update Question Paper() {
char paper ID[20]; int found = -1;
printf("\n--- UPDATE QUESTION PAPER ---\n");
if(paper Count == 0) {
printf("No question papers available!\n");
return; }
View All Question Papers();
printf("\n Enter Paper ID to update: ");
F gets(paper ID, size of (paper ID), stdin);
Paper ID [strcspn(paper ID, "\n")] = 0;
for(int i = 0; i < paper Count; i++) {
if(strcmp(papers[i].paper ID, paper ID) == 0) {
found = i;
break; } }
```

```
if(found == -1) {
printf("\n✗ Paper ID not found!\n");
return; }

printf("\n--- Current Details ---\n");
printf("Subject: %s\n", papers[found].subject);
printf("Year: %d\n", papers[found].year);
printf("Semester: %s\n", papers[found].semester);
printf("Department: %s\n", papers[found].department);

printf("\n--- Enter New Details ---\n");
printf("Enter new Subject Name: ");
fgets(papers[found].subject, size of(papers[found].subject), stdin); papers[found].subject[strcspn(papers[found].subject, "\n")]
= 0;

printf("Enter new Year: ");
scanf("%d", &papers[found].year);
Clear Input Buffer();
```

```
papers[found].semester[strcspn(papers[found].semester, "\n")] = 0;

printf("\n✓ Update Successful!\n");
Save Data();
}
void view All Question Papers() {
printf("\n========================================\n");
printf(" ALL QUESTION PAPERS\n");
printf("========================================\n");
if(paper Count == 0) {
printf("No question papers available!\n");
return; }
printf("%-10s %-20s %-8s %-12s %-15s\n", "Paper ID", "Subject", "Year", "Semester", "Department");
printf("----------------------------------------------------------------------\n");
for(int i = 0; i < paper Count; i++) {
printf("%-10s %-20s %-8d %-12s %-15s\n",
papers[i].paper ID,
papers[i].subject,
papers[i].year,
papers[i].semester,
papers[i].department); }
}
void manage Categories() {
int choice;
printf("\n--- MANAGE CATEGORIES ---\n");
printf("1. Add Subject\n");
printf("2. Delete Subject\n");
printf("3. Add Department\n");
printf("4. Delete Department\n");
printf("5. Back\n");
printf("\n Enter your choice: ");
scanf("%d", &choice);
Clear Input Buffer();
switch(choice) {
case 1:
printf("\n Subject management feature (to be implemented)\n");
break;
case 2:
printf("\n Subject deletion feature (to be implemented)\n");
```

```
case 3:
printf("\n Department management feature (to be implemented)\n");
break;
case 4:
printf("\n Department deletion feature (to be implemented)\n");
break;
case 5:
return;
default:
printf("\n Invalid choice!\n"); }
}

void view Download Statistics() {
printf("\n========================================\n");
printf(" DOWNLOAD STATISTICS\n");
printf("========================================\n");

if(paper Count == 0) {
printf("No question papers available!\n");
return; }
```

```
// Display download count for each paper
printf("\n%-10s %-25s %-15s\n", "Paper ID", "Subject", "Downloads");
printf("------------------------------------------------------\n");
for(int i = 0; i < paper Count; i++) {
printf("%-10s %-25s %-15d\n",
papers[i].paper ID,
papers[i].subject,
papers[i].download Count); }

// Find most downloaded paper
int max Downloads = 0, max Index = 0;
for(int i = 0;
i < paper Count; i++) {
if(papers[i].download Count > max Downloads) {
Max Downloads = papers[i].download Count;
Max Index = i; }
}
if(max Downloads > 0) {
printf("\n🏆 Most Downloaded Paper:\n");
printf("Subject: %s (%d downloads)\n", papers[maxIndex].subject, maxDownloads); } }
```

```
void student Login() {
char rollNumber[20], password[20];
int found = -1;

printf("\n--- STUDENT LOGIN ---\n");
printf("Enter Roll Number: ");
F gets(roll Number, size of(roll Number), stdin);
Roll Number [strcspn(roll Number, "\n")] = 0;
printf("Enter Password: ");
F gets(password, sizeof(password), stdin);
password[strcspn(password, "\n")] = 0;

// Validate credentials
for(int i = 0; i < student Count; i++) {
if(strcmp(students[i].roll Number, roll Number) == 0 && strcmp(students[i].password, password) == 0) {
found = i;
break; } }
if(found != -1) {
printf("\n✓ Login Successful!\n");
printf("Welcome, %s!\n", students[found].name);
Student Menu(roll Number); }
else {
printf("\n✗ Invalid Credentials!\n"); }
}
void student Menu(char *roll Number) {
int choice;
while(1) {
printf("\n========================================\n");
printf(" STUDENT MENU\n");
printf("========================================\n");
printf("1. Search Question Papers\n");
printf("2. Browse by Subject\n");
printf("3. Browse by Year\n");
printf("4. Browse by Department\n");
printf("5. View My Downloads\n");
printf("6. Download Question Paper\n");
printf("7. Logout\n");
printf("========================================\n");
printf("\n Enter your choice: ");
scanf("%d", &choice);
Clear Input Buffer();
switch(choice) {
case 1:
Search Question Papers(roll Number);
break;
case 2:
Browse By Subject(roll Number);
break;
case 3:
Browse By Year(roll Number);
break;
case 4:
Browse By Department(roll Number);
break;
case 5:
View My Downloads(roll Number);
break;
case 6:
Download Question Paper(roll Number);
break;
case 7:
printf("\n Logging out...\n");
return;
default:
printf("\n Invalid choice!\n"); }
}
}
void search Question Papers(char *roll Number) {
char keyword[50], subject[50], semester[20], department[50];
int year;

printf("\n--- SEARCH QUESTION PAPERS ---\n");
printf("Enter search criteria:\n");
printf("Keyword (or press Enter to skip): ");
fgets(keyword, size of(keyword), stdin);
keyword[strcspn(keyword, "\n")] = 0;
```

```
printf("Subject (or press Enter to skip): ");
fgets(subject, size of(subject), stdin);
subject[strcspn(subject, "\n")] = 0;
printf("Year (0 to skip): ");
scanf("%d", &year);
Clear Input Buffer();

printf("Semester (or press Enter to skip): ");
fgets(semester, size of(semester), stdin);
semester[strcspn(semester, "\n")] = 0;

printf("Department (or press Enter to skip): ");
fgets(department, size of(department), stdin);
department[strcspn(department, "\n")] = 0;

printf("\n--- SEARCH RESULTS ---\n");
int found = 0;

for(int i = 0; i < paper Count; i++) {
int match = 1;
if(strlen(keyword) > 0 && strstr(papers[i].subject, keyword) == NULL)
match = 0;
if(strlen(subject) > 0 && strcmp(papers[i].subject, subject) != 0)
match = 0;
if(year > 0 && papers[i].year != year)
match = 0;
if(strlen(semester) > 0 && strcmp(papers[i].semester, semester) != 0)
match = 0;
if(strlen(department) > 0 && strcmp(papers[i].department, department) != 0)
match = 0;
if(match) {
printf("\nPaper ID: %s\n", papers[i].paperID);
printf("Subject: %s\n", papers[i].subject);
printf("Year: %d | Semester: %s\n", papers[i].year, papers[i].semester);
printf("Department: %s\n", papers[i].department);
printf("------------------------------\n");
found++; }
}
```

```
if(found == 0) {
printf("No matching results found!\n"); }
else {
printf("\n Total results: %d\n", found); }
}
void browse By Subject(char *roll Number) {
printf("\n--- BROWSE BY SUBJECT ---\n");
if(paper Count == 0) {
printf("No question papers available!\n");
return; }

// Display unique subjects
char subjects[100][50];
int subject Count = 0;
for(int i = 0; i < paper Count; i++) {
int exists = 0;
for(int j = 0; j < subject Count; j++) {
if(strcmp(subjects[j], papers[i].subject) == 0) {
exists = 1;
break; } }
```

```
if(!exists) {
strcpy(subjects[subject Count++],
papers[i].subject); }
}
printf("\n Available Subjects:\n");
for(int i = 0; i < subject Count; i++) {
printf("%d. %s\n", i+1, subjects[i]); }
int choice;
printf("\n Select subject number: ");
scanf("%d", &choice);
Clear Input Buffer();

if(choice < 1 || choice > subject Count) {
printf("Invalid choice!\n");
return; }

printf("\n--- Papers for %s ---\n", subjects[choice-1]);
for(int i = 0; i < paper Count; i++) {
if(strcmp(papers[i].subject, subjects[choice-1]) == 0)
{
```

```
printf("\n Paper ID: %s\n", papers[i].paper ID);
printf("Year: %d | Semester: %s\n", papers[i].year, papers[i].semester);
printf("--------------------------------\n"); }
}
}
void browse By Year(char *roll Number) {
printf("\n--- BROWSE BY YEAR ---\n");
if(paper Count == 0) {
printf("No question papers available!\n");
return; }

// Display unique years
int years[50], year Count = 0;
for(int i = 0; i < paper Count; i++) {
int exists = 0;
for(int j = 0; j < year Count; j++) {
if(years[j] == papers[i].year) {
exists = 1;
break; }
}
```

```
if(!exists) {
years[year Count++] = papers[i].year; }
}
printf("\n Available Years:\n");
for(int i = 0; i < year Count; i++) {
printf("%d. %d\n", i+1, years[i]);
}
int choice;
printf("\n Select year number: ");
scanf("%d", &choice);
Clear Input Buffer();
if(choice < 1 || choice > year Count) {
printf("Invalid choice!\n");
return; }
printf("\n--- Papers for year %d ---\n", years[choice-1]);
for(int i = 0; i < paper Count; i++) {
if(papers[i].year == years[choice-1]) {
printf("\n Paper ID: %s\n", papers[i].paper ID);
printf("Subject: %s | Semester: %s\n", papers[i].subject, papers[i].semester);
printf("------------------------------\n"); }
}
}
void browse By Department(char *roll Number) {
printf("\n--- BROWSE BY DEPARTMENT ---\n");
if(paper Count == 0) {
Printf ("No question papers available!\n");
return; }
char departments[50][50];
int dept Count = 0;
for(int i = 0; i < paper Count; i++) {
int exists = 0;
for(int j = 0; j < dept Count; j++) {
if(strcmp(departments[j], papers[i].department) == 0) {
exists = 1;
break; } }
if(!exists) {
strcpy(departments[dept Count++], papers[i].department);
} }
```

```
printf("\n Available Departments:\n");
for(int i = 0; i < dept Count; i++) {
printf("%d. %s\n", i+1, departments[i]); }
int choice;
printf("\n Select department number: ");
scanf("%d", &choice);
Clear Input Buffer();
if(choice < 1 || choice > dept Count) {
printf("Invalid choice!\n");
return; }
printf("\n--- Papers for %s ---\n", departments[choice-1]);
for(int i = 0; i < paper Count; i++) {
if(strcmp(papers[i].department, departments[choice-1]) == 0) {
printf("\n Paper ID: %s\n", papers[i].paper ID);
printf("Subject: %s | Year: %d\n", papers[i].subject, papers[i].year);
printf("------------------------------\n"); }
}
}
```

```
void view My Downloads(char *roll Number) {
printf("\n--- MY DOWNLOADS ---\n");
int found = 0;
for(int i = 0; i < down load Count; i++) {
if(strcmp(downloads[i].roll Number, roll Number) == 0) {
printf("\n Paper ID: %s\n", downloads[i].paperID);
printf("Download Date: %s\n", downloads[i].download Date);
printf("--------------------------------\n");
found++; }
}
if(found == 0) {
printf("No download history found!\n"); }
else {
printf("\n Total downloads: %d\n", found); }
}
void download Question Paper(char *roll Number) {
char paper ID[20];
int found = -1;
printf("\n--- DOWNLOAD QUESTION PAPER ---\n");
```

```
View All Question Papers();
printf("\n Enter Paper ID to download: ");
fgets(paper ID, size of(paper ID), stdin);
Paper ID[strcspn(paper ID, "\n")] = 0;
for(int i = 0; i < paper Count; i++) {
if(strcmp(papers[i].paper ID, paper ID) == 0) {
found = i;
break; }
}
if(found == -1) {
printf("\n✗ Paper ID not found!\n");
return; }

// Increment download counter
papers[found].download Count++;

// Record download in student history
strcpy(downloads[download Count].roll Number, roll Number); strcpy(downloads[download Count].paper ID, paper ID);
```

```
Get Current Date(downloads[download Count].download Date);
Download Count++;
printf("\n✓ Download Successful!\n");
printf("File: %s\n", papers[found].filename);
Save Data();}
void register New Student() {
Student new student;
printf("\n--- REGISTER NEW STUDENT ---\n");
printf("Enter Name: ");
fgets(newStudent.name, size of(newStudent.name), stdin);
newStudent.name[strcspn(newStudent.name, "\n")] = 0;
printf("Enter Roll Number: ");
fgets(new Student . rollNumber, size of(new Student.roll Number), stdin);
newStudent. Roll Number[strcspn(new Student.roll Number, "\n")] = 0;

// Check if roll number already exists
for(int i = 0; i < student Count; i++) {
if(strcmp(students[i].rollNumber, newStudent.rollNumber) == 0) {
printf("\n✗ Already Registered!\n");
printf("This roll number is already in the system.\n");
return; }
}
printf("Enter Department: ");
fgets(newStudent.department, sizeof(newStudent.department), stdin);
newStudent.department[strcspn(newStudent.department, "\n")] = 0;
printf("Enter Semester: ");
fgets(newStudent.semester, sizeof(newStudent.semester), stdin); newStudent.semester[strcspn(newStudent.semester,
"\n")] = 0;
printf("Enter Email: ");
fgets(newStudent.email, sizeof(newStudent.email), stdin);
newStudent.email[strcspn(newStudent.email, "\n")] = 0;

printf("Enter Password: ");
fgets(newStudent.password, sizeof(newStudent.password), stdin); newStudent.password[strcspn(newStudent.password,
"\n")] = 0;

// Validate all fields
if(strlen(newStudent.name) == 0 ||
strlen(newStudent.rollNumber) == 0 ||
strlen(newStudent.password) == 0) {
printf("\n✗ All fields are required!\n");
return; }

// Store student details in database
students[studentCount++] = newStudent;
printf("\n✓ Registration Successful!\n");
printf("You can now login with your roll number and password.\n");
saveData(); }

void generatePaperID(char *id) {
sprintf(id, "QP%d", paperCount + 1001);}
void getCurrentDate(char *date) {
sprintf(date, "05-12-2025");
// Simplified date
}

void clearInputBuffer() {
int c;
while ((c = getchar()) != '\n' && c != EOF);
```

```
strlen(newStudent.password) == 0) {
printf("\n✗ All fields are required!\n");
return; }

// Store student details in database
students[studentCount++] = newStudent;
printf("\n✓ Registration Successful!\n");
printf("You can now login with your roll number and password.\n");
saveData(); }

void generatePaperID(char *id) {
sprintf(id, "QP%d", paperCount + 1001);}
void getCurrentDate(char *date) {
sprintf(date, "05-12-2025");
// Simplified date
}

void clearInputBuffer() {
int c;
while ((c = getchar()) != '\n' && c != EOF);
```

```
void loadData() {
FILE *fp;

// Load papers fp = fopen("papers.dat", "rb");
if(fp != NULL) {
fread(&paperCount, sizeof(int), 1, fp);
fread(papers, sizeof(QuestionPaper), paperCount, fp);
fclose(fp); }

// Load students fp = fopen("students.dat", "rb");
if(fp != NULL) {
fread(&studentCount, sizeof(int), 1, fp);
fread(students, sizeof(Student), studentCount, fp);
fclose(fp); }

// Load downloads
fp = fopen("downloads.dat", "rb");
if(fp != NULL) {
fread(&downloadCount, sizeof(int), 1, fp);
fread(downloads, sizeof(DownloadHistory), downloadCount, fp);
fclose(fp); }
```

```
void saveData() {
FILE *fp;

// Save papers
fp = fopen("papers.dat", "wb");
if(fp != NULL) {
fwrite(&paperCount, sizeof(int), 1, fp);
fwrite(papers, sizeof(QuestionPaper), paperCount, fp);
fclose(fp); }

// Save students fp = fopen("students.dat", "wb");
if(fp != NULL) {
fwrite(&studentCount, sizeof(int), 1, fp);
fwrite(students, sizeof(Student), studentCount, fp);
fclose(fp); }

// Save downloads fp = fopen("downloads.dat", "wb");
if(fp != NULL) {
fwrite(&downloadCount, sizeof(int), 1, fp);
fwrite(downloads, sizeof(DownloadHistory), downloadCount, fp);
fclose(fp);}
}
```

OUTPUT :

```
==========================================
  QUESTION PAPERS MANAGEMENT SYSTEM
==========================================
1. Admin Login
2. Student Login
3. Register New Student
4. Exit
==========================================

Enter your choice:1
```

```
--- ADMIN LOGIN ---
Enter username: admin
Enter password: admin123

? Login Successful!

========================================
  ADMIN MENU (Logged in as: admin)
========================================
1. Upload Question Paper
2. Delete Question Paper
3. Update Question Paper
4. View All Question Papers
5. Manage Categories/Subjects
6. View Download Statistics
7. Logout
========================================

Enter your choice:1
```

```
--- UPLOAD QUESTION PAPER ---
Enter Subject Name: maths
Enter Year: 2025
Enter Semester: 1
Enter Exam Type (Mid/Final/Model): mid
Enter Department: civil
Enter Regulation: 1
Enter PDF filename: maths.pdf

? Upload Successful!
Paper ID: QP1002

========================================
  ADMIN MENU (Logged in as: admin)
========================================
1. Upload Question Paper
2. Delete Question Paper
3. Update Question Paper
4. View All Question Papers
5. Manage Categories/Subjects
6. View Download Statistics
7. Logout
```

```
==========================================

Enter your choice: 2

--- DELETE QUESTION PAPER ---

==========================================
  ALL QUESTION PAPERS
==========================================
Paper ID   Subject            Year    Semester    Department
----------------------------------------------------------------------
QP1001     MATHS               2024    1           CIVIL
QP1002     maths              2025    1           civil

Enter Paper ID to delete: QP1001

Are you sure you want to delete this paper? (y/n): Y

? Deletion Success
==========================================
  ADMIN MENU (Logged in as: admin)
==========================================
1. Upload Question Paper
2. Delete Question Paper
3. Update Question Paper
4. View All Question Papers
5. Manage Categories/Subjects
6. View Download Statistics
7. Logout
==========================================

Enter your choice: 3

--- UPDATE QUESTION PAPER ---

==========================================
  ALL QUESTION PAPERS
==========================================
Paper ID   Subject            Year     Semester     Department
------------------------------------------------------------------------
QP1002     maths              2025     1            civil

Enter Paper ID to update: QP1002
```

```
--- Current Details ---
Subject: maths
Year: 2025
Semester: 1
Department: civil

--- Enter New Details ---
Enter new Subject Name: PHYSICS
Enter new Year: 2025
Enter new Semester: 1

? Update Successful!

========================================
  ADMIN MENU (Logged in as: admin)
========================================
1. Upload Question Paper
2. Delete Question Paper
3. Update Question Paper
4. View All Question Papers
5. Manage Categories/Subjec
```

```
6. View Download Statistics
7. Logout
==========================================

Enter your choice:

==========================================
  ALL QUESTION PAPERS
==========================================
Paper ID   Subject            Year     Semester     Department
----------------------------------------------------------------------
QP1002     PHYSICS            2025     1            civil

==========================================
  ADMIN MENU (Logged in as: admin)
==========================================
1. Upload Question Paper
2. Delete Question Paper
3. Update Question Paper
4. View All Question Papers
5. Manage Categories/Subjects
6. View Download Statistics
```

```
7. Logout
========================================
Enter your choice:5
--- MANAGE CATEGORIES ---
1. Add Subject
2. Delete Subject
3. Add Department
4. Delete Department
5. Back
Enter your choice: 1

Subject management feature (to be implemented)
========================================
  ADMIN MENU (Logged in as: admin)
========================================
1. Upload Question Paper
2. Delete Question Paper
3. Update Question Paper
4. View All Question Papers
5. Manage Categories/Subjects
6. View Download Statistics
7. Logout
```

```
============================================

Enter your choice: 6

============================================
  DOWNLOAD STATISTICS
============================================

Paper ID   Subject                 Downloads
------------------------------------------------------
QP1002     PHYSICS                   0

============================================
  ADMIN MENU (Logged in as: admin)
============================================
1. Upload Question Paper
2. Delete Question Paper
3. Update Question Paper
4. View All Question Papers
5. Manage Categories/Subjects
6. View Download Statistics
7. Logout
```

```
========================================

Enter your choice: 7

Logging out...

========================================
  QUESTION PAPERS MANAGEMENT SYSTEM
========================================
1. Admin Login
2. Student Login
3. Register New Student
4. Exit
========================================

Enter your choice: 2

--- STUDENT LOGIN --
```

```
Enter your choice: 3

--- REGISTER NEW STUDENT ---
Enter Name: jaya sri
Enter Roll Number: newstudent.rollnumber
Enter Department: Enter Semester: civil 1
Enter Email: jayasri2259@gmail.com
Enter Password: js12@@js

? Registration Successful!
You can now login with your roll number and password.

========================================
  QUESTION PAPERS MANAGEMENT SYSTEM
========================================
1. Admin Login
2. Student Login
3. Register New Student
4. Exit
========================================

Enter your choice:
```

```
Enter your choice: 4

Thank you for using Question Papers Management System!

--------------------------------
Process exited after 2708 seconds with return value 0
Press any key to continue . . .
```

CONCLUSION :

The Previous Year Question Papers Management System successfully provides a simple, fast, and efficient way to store, manage, and retrieve question papers using basic C programming concepts. By integrating modules such as data input, file reading, storage, and searching, the system ensures that users can easily upload question papers and access them whenever required . This project demonstrates the practical use of arrays, strings, functions, file handling, and menu-driven programming. It reduces manual effort, eliminates the risk of misplaced papers, and saves time by providing quick search results based on date, subject, and exam type.

Overall, the system achieves its goal of creating an organized digital repository for previous year question papers and can be further improved by adding features like editing, deleting, or storing data in external databases.

THANK YOU