PROJECT REPORT

ON

ONLINE EXAMINATION PORTAL

Submitted

By

P. RAJESH BABU – U202405204810356

BATCH – PGDEA 83

DEPARTMENT OF PG DIPLOMA IN EMBEDDED AND AUTOMOTIVE SYSTEMS

CRANES VARSITY

BANGALORE, KARNATAKA, INDIA - 560001

AUGUST - 2024

TABLE OF CONTENTS

ABSTRACT

The online examination portal is a web-based application designed to streamline the examination process for educational institutions. Traditional examination methods, often burdened by logistical challenges, inefficiencies, and the potential for human error, have driven the need for a more efficient and accessible system. This project aims to address these issues by providing a comprehensive platform that allows educators to create, administer, and evaluate exams online. The portal supports multiple types of questions, including multiple choice, true/false, and descriptive formats, ensuring a versatile assessment environment. It features secure user authentication, automated grading for objective questions, and real-time result processing, significantly reducing the time and resources required for examination administration.

INTRODUCTION

This C-based program implements a simple yet functional online examination portal designed to manage user registrations, login, exam creation, and exam-taking activities. It leverages basic file handling for persistent storage of user information and exams, ensuring that data remains accessible across multiple sessions. The portal is structured to cater to both administrators and regular users, with distinct functionalities tailored to each role.

In this system, the user and exam data are stored persistently using both text files and binary files. This dual approach allows the program to save and load user credentials and exam details, ensuring that data is retained between program executions. The user-related functionalities include registration, login, and taking exams, while the administrator is provided with the capability to create new exams.

User registration involves entering a username and password, which are then saved in both a text file and a binary file for future reference. The program ensures that each username is unique by checking against existing records before saving the new user. This ensures the integrity and security of the registration process.

The exam creation process is handled by the administrator, who can define the exam name, number of questions, and the available options for each question. The correct answers are also recorded during this process. The exams are saved persistently in both text and binary formats, allowing them to be retrieved and taken by users at any time.

Once registered and logged in, users can select from available exams, answer the questions, and receive their scores immediately after completing the exam. This interactive experience, though simple, effectively mimics the functionality of a basic online examination system.

Overall, this C program offers a foundational example of how to manage users and exams in a command-line environment, demonstrating essential programming techniques such as file handling, struct usage, and basic input/output operations. It serves as an excellent starting point for anyone interested in developing more complex systems with similar functionalities.

OBJECTIVE

The primary objective of this project is to create a simple and efficient online examination portal that facilitates user registration, secure login, exam creation, and exam participation within a command-line interface. By implementing core functionalities such as persistent data storage, user authentication, and interactive exam-taking, the program aims to provide a foundational platform for managing and conducting exams in a digital environment. This project is designed to demonstrate key programming concepts, including file handling for data persistence, struct-based data management, and user input processing. The goal is to offer a practical example of how to integrate these elements into a cohesive system that can serve as a stepping stone for more advanced online examination systems.

FLOWCHART

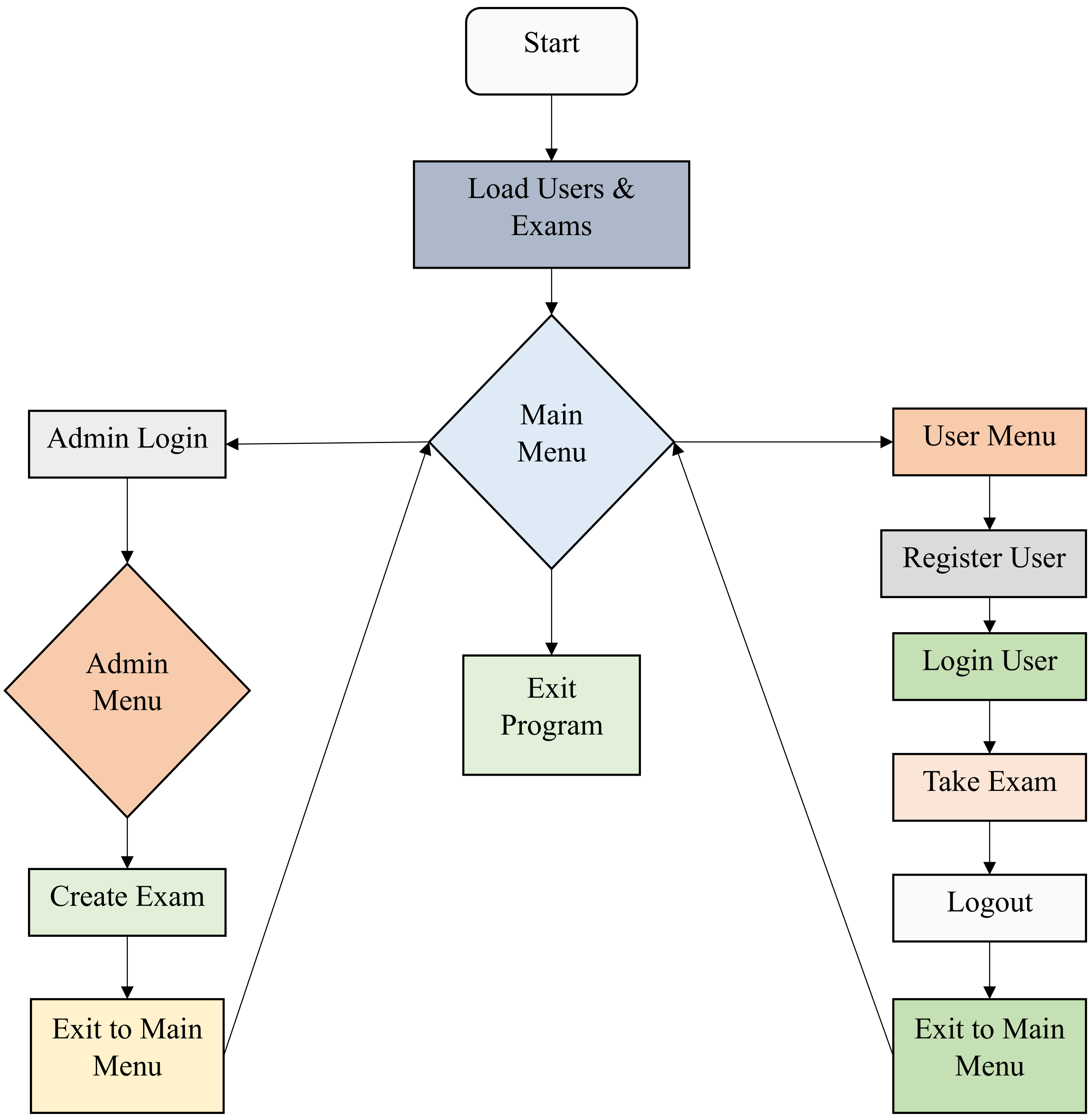


Fig1: Flow Chart of Online Examination Portal

USE CASE DIAGRAM

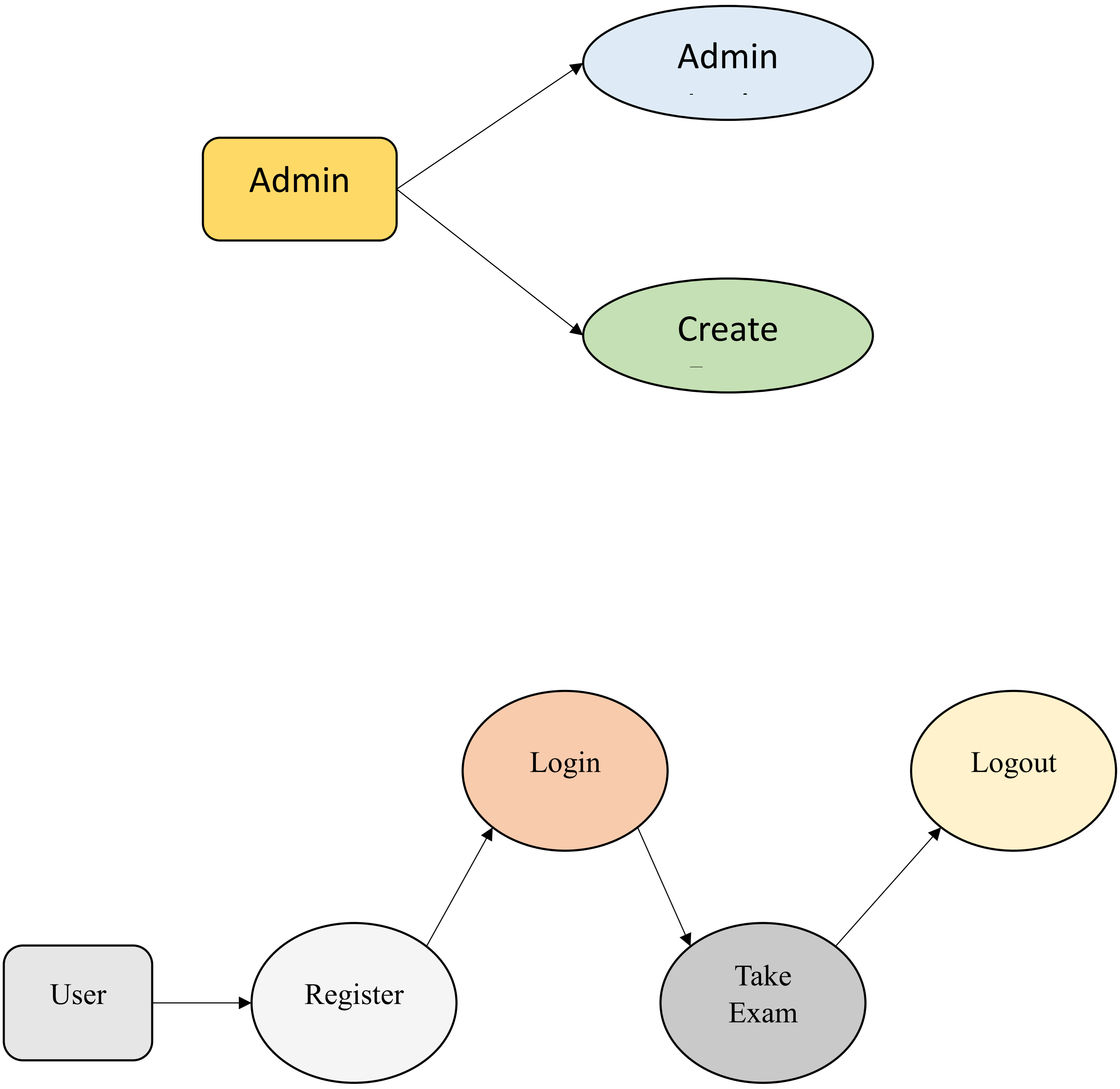


Fig2: Use Case Diagram of Online Examination Portal

IMPLEMENTATION & FUNCTIONALITIES

Implementation:

I implement this project using file handling functions like fopen (), fclose (), fgets (), fputs () etc...

Functionalities:

1. Data Structures

- **User**: Stores information about users, including their username and password.
- **Question**: Represents a question in an exam with its options and the correct answer.
- **Exam**: Contains details about an exam, including its name, a list of questions, and the number of questions.

2. Data Management Functions

- **saveUsersToText**: Saves user data to a text file (users.txt).
- **loadUsersFromText**: Loads user data from the text file (users.txt) and updates the user's array.
- **loadUsers**: Loads user data from both text (users.txt) and binary (users.dat) files.
- **saveUsers**: Saves user data to both text (users.txt) and binary (users.dat) files.
- **saveExamsToText**: Saves exam data to a text file (exams.txt).
- **loadExamsFromText**: Loads exam data from the text file (exams.txt) and updates the exams array.
- **loadExams**: Loads exam data from both text (exams.txt) and binary (exams.dat) files.

- **saveExams**: Saves exam data to both text (exams.txt) and binary (exams.dat) files.

3. User Registration and Login

- **registerUser**: Allows a new user to register by entering a username and password, checking for existing usernames, and saving the user data.
- **loginUser**: Allows an existing user to log in by entering their username and password, verifying them against stored data.

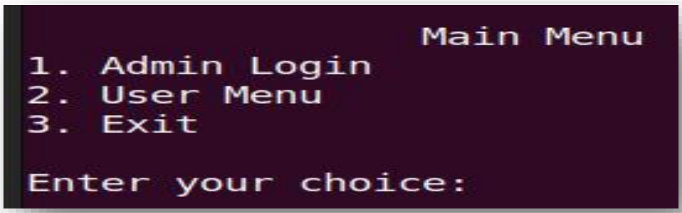
4. Exam Management

- **createExam**: Enables an admin to create a new exam by providing an exam name, the number of questions, each question with its options, and the correct answer.
- **takeExam**: Allows a logged-in user to take an exam by providing the exam name, displaying questions and options, and collecting user answers to calculate the score.

5. Menus and Program Flow

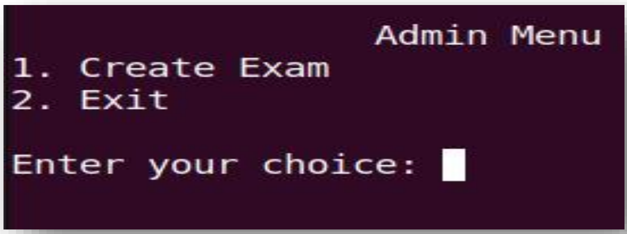
- **adminMenu**: Provides the admin with options to create an exam or exit the admin menu.
- **userMenu**: Provides users with options to register, log in, take an exam, or log out.
- **main**: Displays the main menu with options for admin login, user menu, or exit. It manages the overall flow of the program by calling appropriate menu functions.

RESULTS



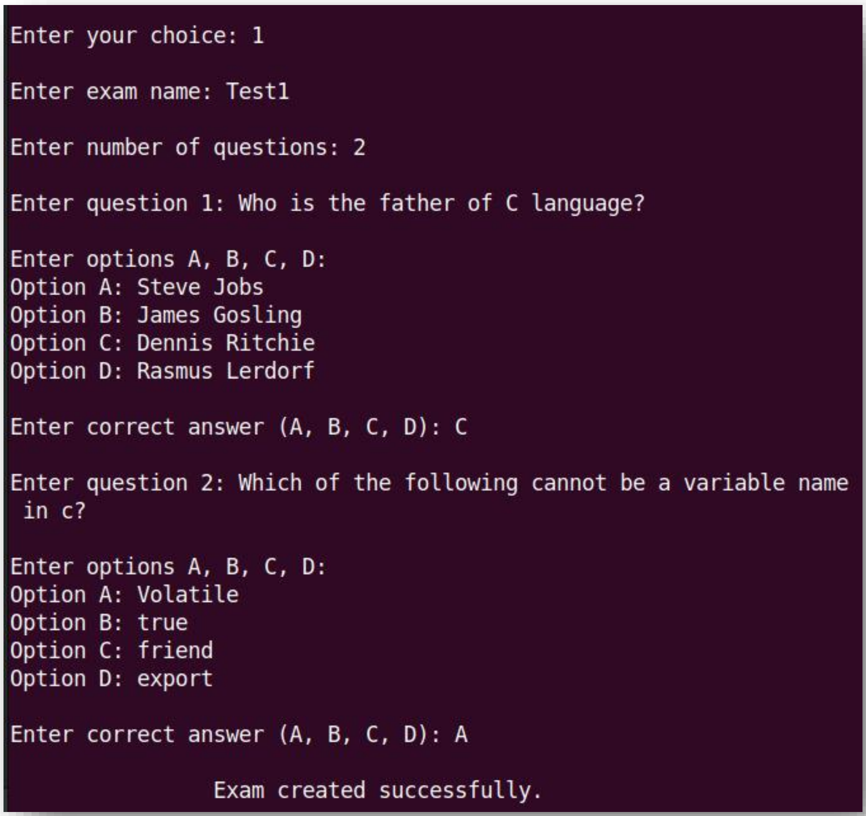
```
                         Main Menu
1. Admin Login
2. User Menu
3. Exit

Enter your choice:
```

Fig3: Main menu



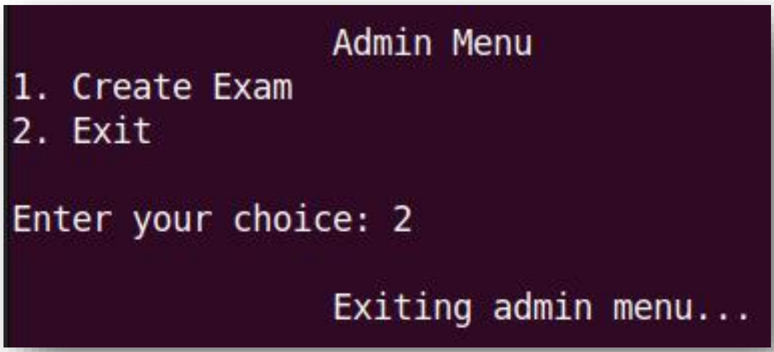
```
                         Admin Menu
1. Create Exam
2. Exit

Enter your choice: █
```

Fig4: Admin Menu



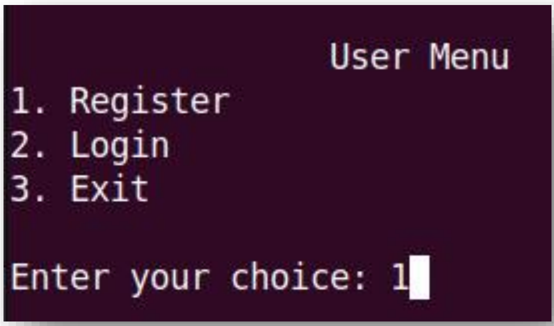
```
Enter your choice: 1

Enter exam name: Test1

Enter number of questions: 2

Enter question 1: Who is the father of C language?

Enter options A, B, C, D:
Option A: Steve Jobs
Option B: James Gosling
Option C: Dennis Ritchie
Option D: Rasmus Lerdorf

Enter correct answer (A, B, C, D): C

Enter question 2: Which of the following cannot be a variable name
 in c?

Enter options A, B, C, D:
Option A: Volatile
Option B: true
Option C: friend
Option D: export

Enter correct answer (A, B, C, D): A

                Exam created successfully.
```

Fig5: Exam Creation



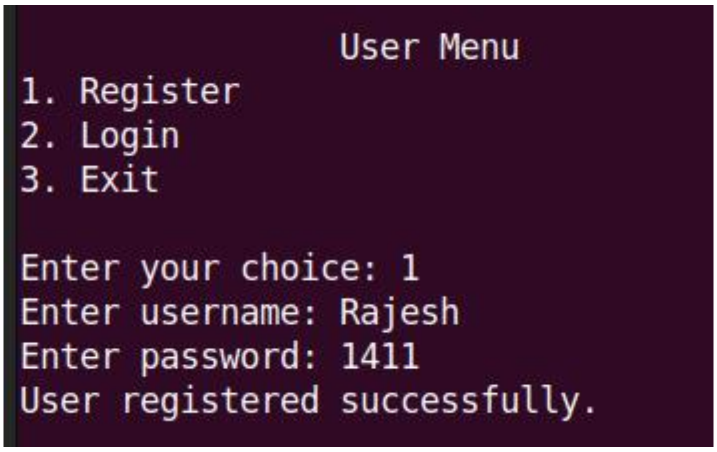
```
                Admin Menu
1. Create Exam
2. Exit

Enter your choice: 2

                Exiting admin menu...
```

Fig6: Exit From Admin Menu



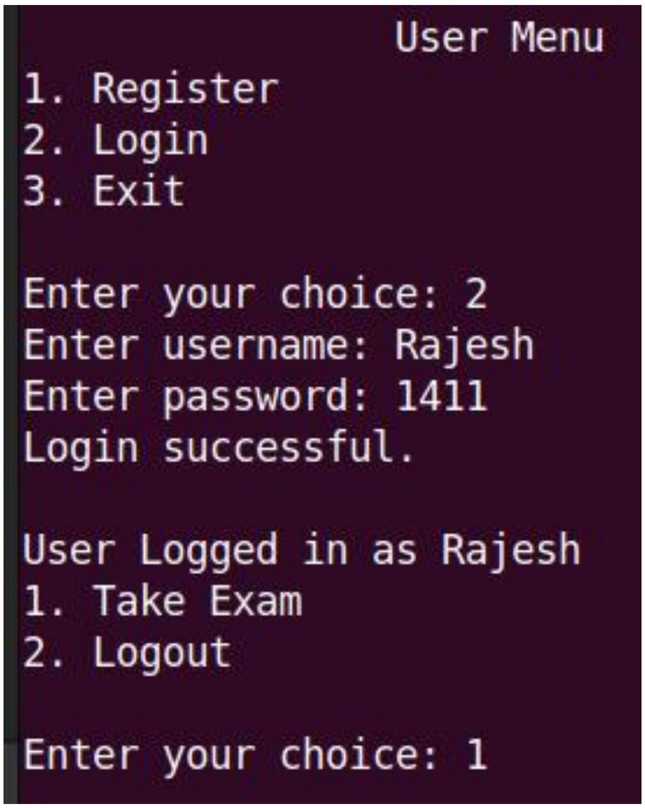
```
                    User Menu
1. Register
2. Login
3. Exit

Enter your choice: 1█
```

Fig7: User Menu



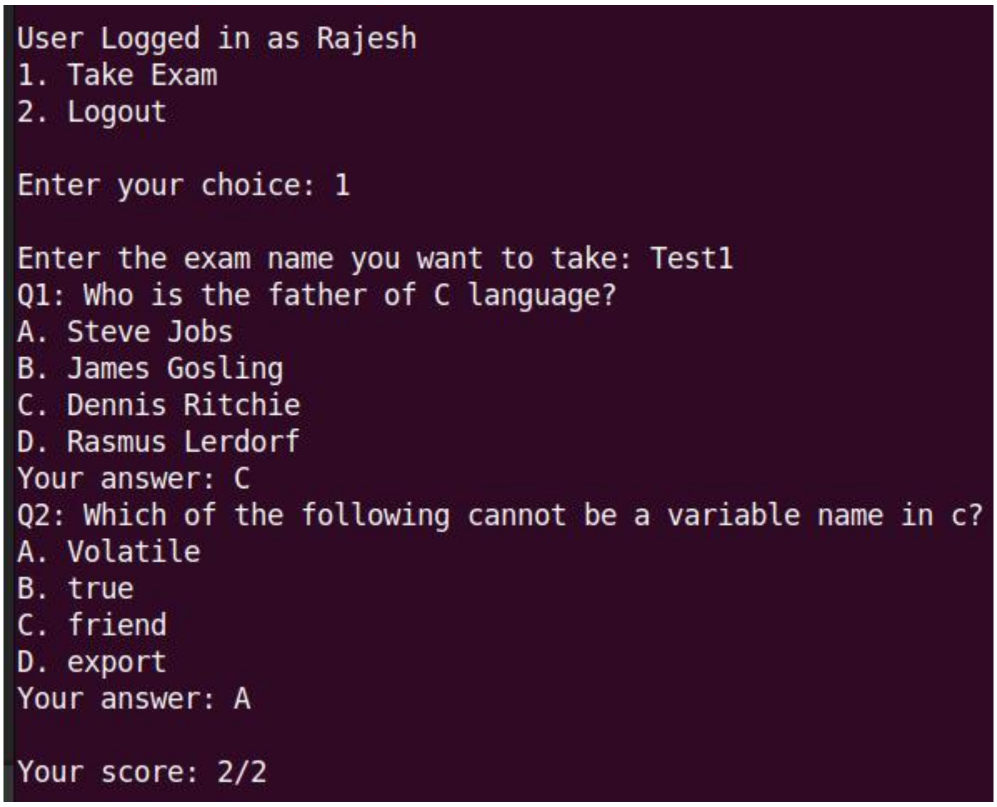
```
                 User Menu
1. Register
2. Login
3. Exit

Enter your choice: 1
Enter username: Rajesh
Enter password: 1411
User registered successfully.
```

Fig8: User Register



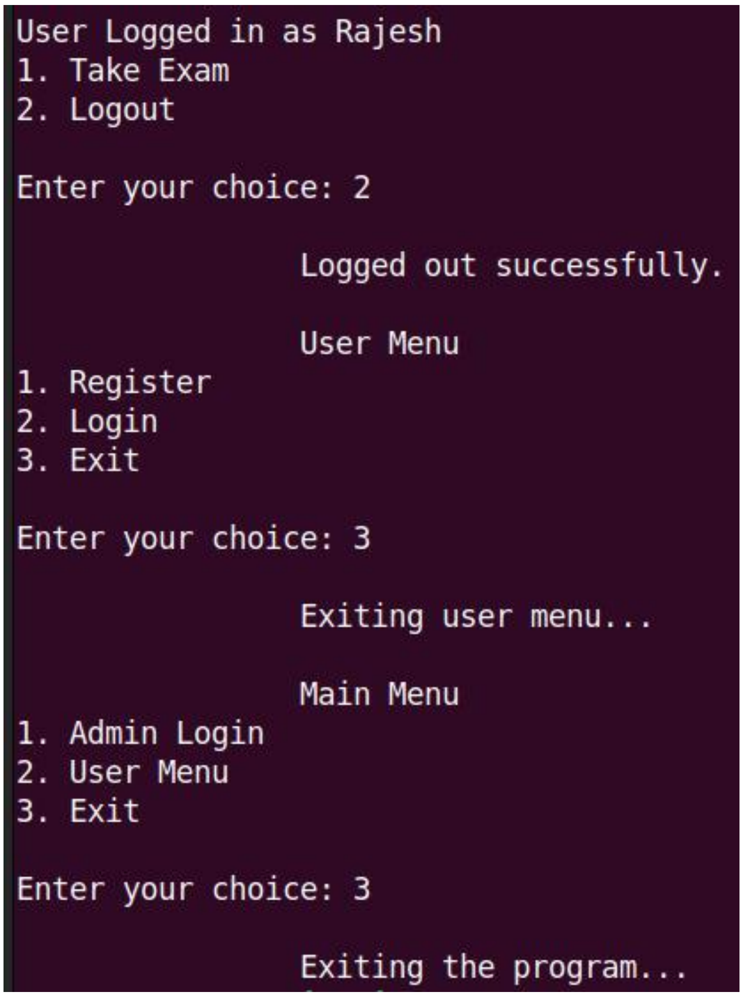
```
                 User Menu
1. Register
2. Login
3. Exit

Enter your choice: 2
Enter username: Rajesh
Enter password: 1411
Login successful.

User Logged in as Rajesh
1. Take Exam
2. Logout

Enter your choice: 1
```

Fig9: User Login



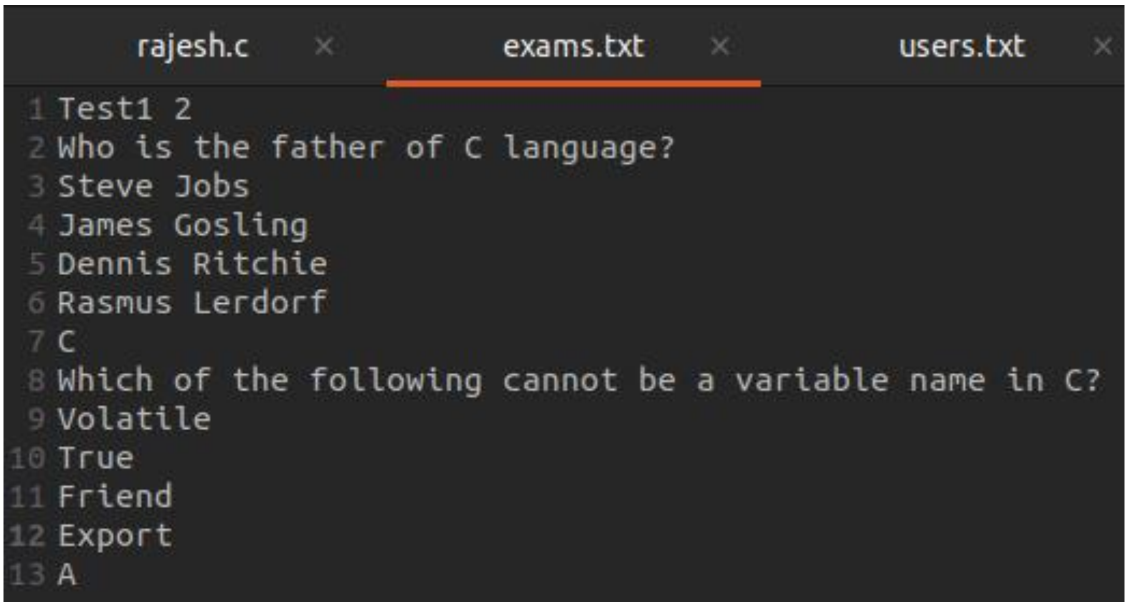
```
User Logged in as Rajesh
1. Take Exam
2. Logout

Enter your choice: 1

Enter the exam name you want to take: Test1
Q1: Who is the father of C language?
A. Steve Jobs
B. James Gosling
C. Dennis Ritchie
D. Rasmus Lerdorf
Your answer: C
Q2: Which of the following cannot be a variable name in c?
A. Volatile
B. true
C. friend
D. export
Your answer: A

Your score: 2/2
```

Fig10: User Take Exam



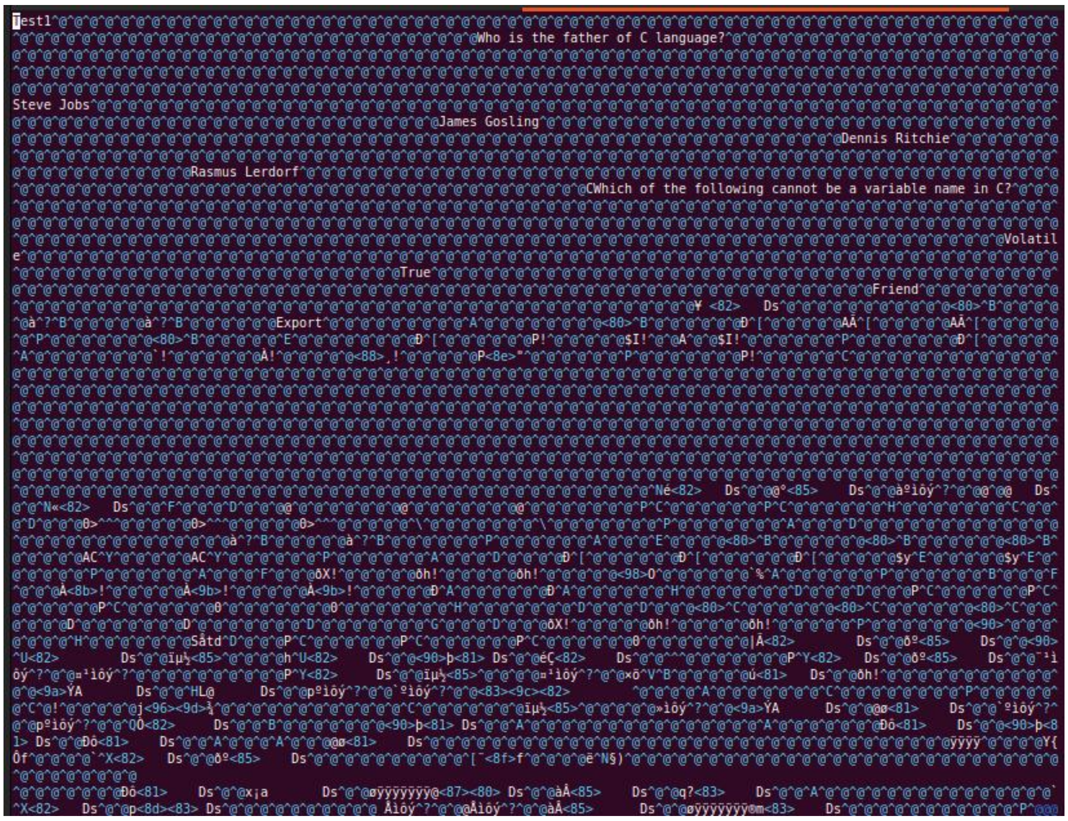
```
User Logged in as Rajesh
1. Take Exam
2. Logout

Enter your choice: 2

                Logged out successfully.

                User Menu
1. Register
2. Login
3. Exit

Enter your choice: 3

                Exiting user menu...

                Main Menu
1. Admin Login
2. User Menu
3. Exit

Enter your choice: 3

                Exiting the program...
```

Fig11: User Logout and Exit from Program



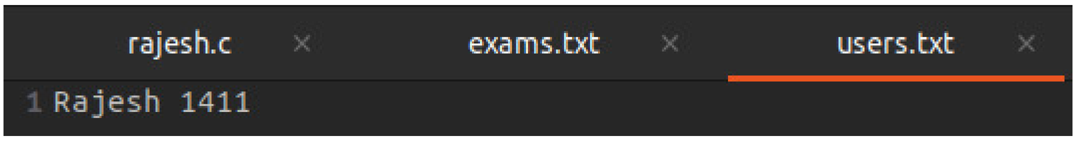
```
Test1 2
Who is the father of C language?
Steve Jobs
James Gosling
Dennis Ritchie
Rasmus Lerdorf
C
Which of the following cannot be a variable name in C?
Volatile
True
Friend
Export
A
```

Fig12: Exam Data Stored in Text File



```
Testl^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@Who is the father of C language?^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
Steve Jobs^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@James Gosling^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@Dennis Ritchie^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@@Rasmus Lerdorf^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@CWhich of the following cannot be a variable name in C?^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@Volatil
e^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@True^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@Friend^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@¥ <82>    Ds^@^@^@^@^@^@^@^@^@^@^@^@<80>^B^@^@^@
^@à^?^B^@^@^@^@à^?^B^@^@^@^@^@Export^@^@^@^@^@^@^@^@A^@^@^@^@^@^@^@<80>^B^@^@^@^@^@Ð^[^@^@^@^@AÅ^[^@^@^@^@AÅ^[^@^@^@
^@^P^@^@^@^@^@^@<80>^B^@^@^@^@^E^@^@^@^@^@^@^@Ð^[^@^@^@^@^@P!^@^@^@^@$I!^@^@A^@^@$I!^@^@^@^@^@^@^P^@^@^@^@^@^@Ð^[^@^@^@
^A^@^@^@^@^@^@^@`!^@^@^@^@^@À!^@^@^@^@^@<88>¸!^@^@^@^@P<8e>"^@^@^@^@^@^P^@^@^@^@^@^@P!^@^@^@^@^C^@^@^@^@^@^@^@^@^@^@^@^@
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^Né<82>    Ds^@^@@°<85>    Ds^@^@àºìôý^?^@^@@^@@   Ds^
@^@^N«<82>   Ds^@^@^F^@^@^@^D^@^@^@@^@^@^@^@^@^@@^@^@^@^@^@^@@^@^@^@^@^@^@^P^C^@^@^@^@^@^@^P^C^@^@^@^@^@^@^H^@^@^@^@^@^@^C^@^@^
@^D^@^@^@@0>^^^@^@^@^@^@0>^^^@^@^@^@@0>^^^@^@^@^@^@\^@^@^@^@^@^@^@\^@^@^@^@^@^@^@P^@^@^@^@^@^@^@A^@^@^@D^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@à^?^B^@^@^@^@^@à^?^B^@^@^@^@^@^@^P^@^@^@^@^@^@^A^@^@^@^E^@^@^@^@<80>^B^@^@^@^@^@^@<80>^B^@^@^@^@^@^@<80>^B^
@^@^@^@^@AC^Y^@^@^@^@^@AC^Y^@^@^@^@^@^@^P^@^@^@^@^@^@A^@^@^@^D^@^@^@^@Ð^[^@^@^@^@^@^@Ð^[^@^@^@^@^@^@Ð^[^@^@^@^@^@$y^E^@^@^@^@^@$y^E^@^
@^@^@^@^@^P^@^@^@^@^@^@^A^@^@^@^F^@^@^@ðX!^@^@^@^@^@ðh!^@^@^@^@^@ðh!^@^@^@^@^@<98>0^@^@^@^@^@^@`%^A^@^@^@^@^@^@^P^@^@^@^@^@^@^B^@^@^@^F
^@^@^@À<8b>!^@^@^@^@^@À<9b>!^@^@^@^@^@À<9b>!^@^@^@^@^@Ð^A^@^@^@^@^@^@Ð^A^@^@^@^@^@^@^H^@^@^@^@^@^@^D^@^@^@^D^@^@^@P^C^@^@^@^@^@^@P^C^
@^@^@^@^@^@P^C^@^@^@^@^@^@0^@^@^@^@^@^@^@0^@^@^@^@^@^@^H^@^@^@^@^@^@^D^@^@^@^D^@^@^@<80>^C^@^@^@^@^@^@<80>^C^@^@^@^@^@^@<80>^C^@^@^
@^@^@^@D^@^@^@^@^@^@^@D^@^@^@^@^@^@^@D^@^@^@^@^@^@^G^@^@^@^D^@^@^@ðX!^@^@^@^@^@ðh!^@^@^@^@^@ðh!^@^@^@^@^@^P^@^@^@^@^@^@<90>^@^@^@^
@^@^@^@^H^@^@^@^@^@^@Såtd^D^@^@^@P^C^@^@^@^@^@^@P^C^@^@^@^@^@^@P^C^@^@^@^@^@^@0^@^@^@^@^@^@^@|Ã<82>        Ds^@^@ðº<85>    Ds^@^@<90>
^U<82>        Ds^@^@ïµ½<85>^@^@^@^@h^U<82>     Ds^@^@<90>þ<81> Ds^@^@éÇ<82>     Ds^@^@^^^@^@^@^@^@^@^@P^Y<82>    Ds^@^@ðº<85>     Ds^@^@¯¹ì
ôý^?^@^@¤¹ìôý^?^@^@^@^@^@^@^@^@^@^@P^Y<82>      Ds^@^@ïµ½<85>^@^@^@^@¤¹ìôý^?^@^@×õ^V^B^@^@^@^@^@ú<81>    Ds^@^@ðh!^@^@^@^@^@^@^@^@^@^@^@^
@^@<9a>ÝA        Ds^@^@^HL@       Ds^@^@pºìôý^?^@^@`ºìôý^?^@^@<83><9c><82>          ^@^@^@^@A^@^@^@^@^@^@^@C^@^@^@^@^@^@^@^@P^@^@^@^@^
@^C^@!^@^@^@^@^@j<96><9d>¾^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^@^@^@^@^@^@ïµ½<85>^@^@^@^@^@»ìôý^?^@^@<9a>ÝA       Ds^@^@@ø<81>     Ds^@^@`ºìôý^?^
@^@pºìôý^?^@^@^QÒ<82>       Ds^@^@^B^@^@^@^@^@^@^@<90>þ<81> Ds^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@Ðô<81>     Ds^@^@<90>þ<8
1> Ds^@^@Ðô<81>     Ds^@^@^A^@^@^@^A^@^@^@@ø<81>     Ds^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@ÿÿÿÿ^@^@^@^@Y{
Ôf^@^@^@^@`^X<82>    Ds^@^@ðº<85>     Ds^@^@^@^@^@^@^@^@^@^@^@^@^@[¯<8f>f^@^@^@^@ë^N§)^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@Ðô<81>     Ds^@^@x¡a        Ds^@^@øÿÿÿÿÿÿÿ@<87><80> Ds^@^@àÅ<85>     Ds^@^@q?<83>     Ds^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@`
^X<82>    Ds^@^@p<8d><83> Ds^@^@^@^@^@^@^@^@^@^@^@^@ Åìôý^?^@^@Åìôý^?^@^@àÅ<85>       Ds^@^@øÿÿÿÿÿÿÿ®m<83>     Ds^@^@^@^@^@^@^@^@^@^@^@^@^@^P^@@@
```

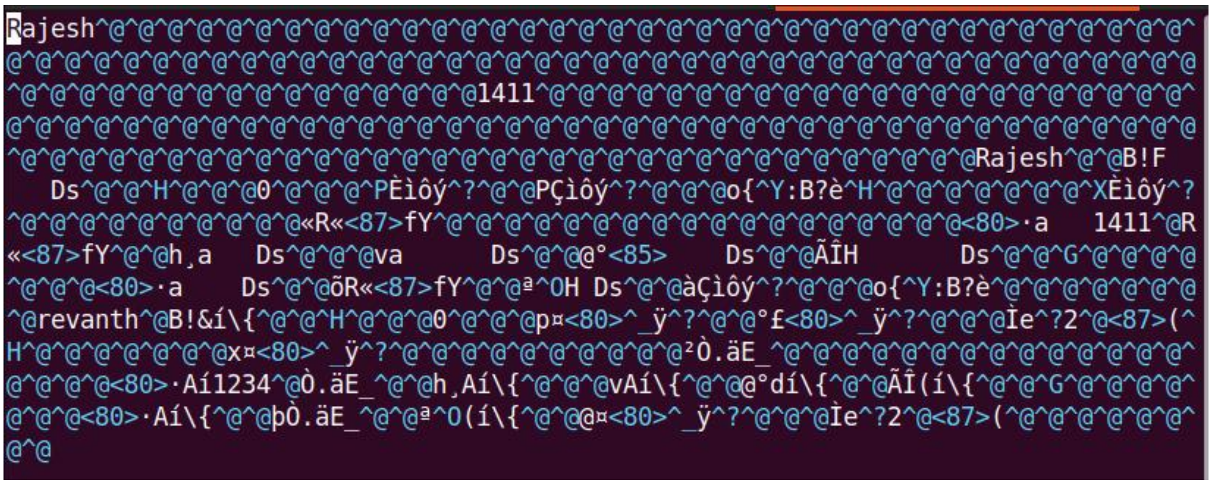
Fig13: Exam Data Stored in Binary File



rajesh.c ×   exams.txt ×   users.txt ×

```
Rajesh 1411
```

Fig14: User Login Data in Text File



```
Rajesh^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@1411^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@Rajesh^@^@B!F
   Ds^@^@^H^@^@^@0^@^@^@^PÈìôý^?^@^@PÇìôý^?^@^@^@o{^Y:B?è^H^@^@^@^@^@^@^@^XÈìôý^?
^@^@^@^@^@^@^@^@^@^@«R«<87>fY^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@<80>·a   1411^@R
«<87>fY^@^@h¸a   Ds^@^@^@va       Ds^@^@@°<85>     Ds^@^@ÃÎH        Ds^@^@^G^@^@^@^@
^@^@^@<80>·a    Ds^@^@õR«<87>fY^@^@ª^OH Ds^@^@àÇìôý^?^@^@^@o{^Y:B?è^@^@^@^@^@^@^@
^@revanth^@B!&í\{^@^@^H^@^@^@0^@^@^@p¤<80>^_ÿ^?^@^@°£<80>^_ÿ^?^@^@^@Ìe^?2^@<87>(^
H^@^@^@^@^@^@^@x¤<80>^_ÿ^?^@^@^@^@^@^@^@^@^@^@²Ò.äE_^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@<80>·Aí1234^@Ò.äE_^@^@h¸Aí\{^@^@^@vAí\{^@^@@°dí\{^@^@ÃÎ(í\{^@^@^G^@^@^@^@^
@^@^@<80>·Aí\{^@^@þÒ.äE_^@^@ª^O(í\{^@^@@¤<80>^_ÿ^?^@^@^@Ìe^?2^@<87>(^@^@^@^@^@^@^
@^@
```

Fig15: User Login Data in Binary File

11

ADVANTAGES

- Convenience And Accessibility
- Time Efficiency
- Security And Integrity
- Time Management
- Environmentally Friendly

APPLICATIONS

- Educational institutions
- Training institutes
- Government and public sector
- Competitive exam coaching
- Online tutoring platforms

CONCLUSION

The Online Examination Portal is to implements a simple exam management system with functionalities for both users and administrators. The system allows users to register, log in, and take exams, while administrators can create and manage exams. The code is structured into several functions that handle different aspects of user and exam management, including loading and saving data from/to files, which ensures that user and exam information persists between sessions.

User registration and login are managed with password protection, while the exam creation process involves entering questions, options, and correct answers. The user's ability to take exams and see their scores offers a basic but effective way to assess knowledge. The implementation uses both text and binary files for data storage, allowing flexibility and redundancy in saving user and exam information.

Overall, this code provides a foundational framework for a simple exam system. It demonstrates the use of file handling in C for persistence, basic user authentication, and dynamic exam management. While there are opportunities for further enhancement, such as adding user feedback or more advanced exam features, the current implementation successfully achieves its core objectives in a clear and manageable manner.

REFERENCES

[1] https://github.com/rakesh-m-r/DBMS-MINI-Project

[2] https://github.com/mk60991/Online-Examination-System/tree/master

[3] https://bsssbhopal.edu.in/uploads/naac/criteria_1/students_projects/156%20Megha%20Paliwal.pdf

[4] https://projectworlds.in/free-projects/php-projects/online-examination/

APPENDIX

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_USERS 100

#define MAX_EXAMS 10

#define MAX_QUESTIONS 10

#define MAX_OPTION_LENGTH 100


typedef struct {

   char question[256];

   char options[4][MAX_OPTION_LENGTH];

   char answer; // 'A', 'B', 'C', 'D'

} Question;


typedef struct {

   char examName[100];

   Question questions[MAX_QUESTIONS];

   int numQuestions;

} Exam;




typedef struct {
    char username[100];

    char password[100];

} User;


User users[MAX_USERS];

Exam exams[MAX_EXAMS];

int userCount = 0;

int examCount = 0;


void saveUsersToText() {

    FILE *file = fopen("users.txt", "w");

    if (file) {

        for (int i = 0; i < userCount; i++) {

            fprintf(file, "%s %s\n", users[i].username, users[i].password);

        }

        fclose(file);

    }

}


void loadUsersFromText() {

    FILE *file = fopen("users.txt", "r");

    if (file) {

        while (fscanf(file, "%s %s", users[userCount].username, users[userCount].password) !=
EOF) {

            userCount++;
        }
        fclose(file);
    }
}

void loadUsers() {
    loadUsersFromText();
    FILE *file = fopen("users.dat", "rb");
    if (file) {
        while (fread(&users[userCount], sizeof(User), 1, file)) {
            userCount++;
        }
        fclose(file);
    }
}

void saveUsers() {
    saveUsersToText();
    FILE *file = fopen("users.dat", "wb");
    if (file) {
        fwrite(users, sizeof(User), userCount, file);
        fclose(file);
    }
}
```

```
int registerUser() {
    if (userCount >= MAX_USERS) {
        printf("User limit reached.\n");
        return 0;
    }

    User newUser;
    printf("\nEnter username: ");
    scanf("%s", newUser.username);
    printf("Enter password: ");
    scanf("%s", newUser.password);

    // Check if username already exists
    for (int i = 0; i < userCount; i++) {
        if (strcmp(users[i].username, newUser.username) == 0) {
            printf("Username already exists.\n");
            return 0;
        }
    }

    users[userCount++] = newUser;
    saveUsers();
    printf("\n\t\tUser registered successfully.\n");
    return 1;
}

int loginUser(char* loggedInUser) {
    char username[100], password[100];
    printf("\nEnter username: ");
    scanf("%s", username);
    printf("Enter password: ");
    scanf("%s", password);

    for (int i = 0; i < userCount; i++) {
        if (strcmp(users[i].username, username) == 0 &&
            strcmp(users[i].password, password) == 0) {
            printf("\n\t\tLogin successful.\n");
            strcpy(loggedInUser, username);
            return 1;
        }
    }

    printf("Invalid username or password.\n");
    return 0;
}

void saveExamsToText() {
    FILE *file = fopen("exams.txt", "w");
    if (file) {
        for (int i = 0; i < examCount; i++) {
            fprintf(file, "%s %d\n", exams[i].examName, exams[i].numQuestions);
            for (int j = 0; j < exams[i].numQuestions; j++) {
                fprintf(file, "%s\n", exams[i].questions[j].question);
                fprintf(file, "%s\n", exams[i].questions[j].options[0]);
                fprintf(file, "%s\n", exams[i].questions[j].options[1]);
                fprintf(file, "%s\n", exams[i].questions[j].options[2]);
                fprintf(file, "%s\n", exams[i].questions[j].options[3]);
                fprintf(file, "%c\n", exams[i].questions[j].answer);
            }
        }
        fclose(file);
    }
}

void loadExamsFromText() {
    FILE *file = fopen("exams.txt", "r");
    if (file) {
        while (fscanf(file, "%s %d", exams[examCount].examName,
&exams[examCount].numQuestions) != EOF) {
            fgetc(file); // Remove newline character after numQuestions
            for (int j = 0; j < exams[examCount].numQuestions; j++) {
                fgets(exams[examCount].questions[j].question, 256, file);
exams[examCount].questions[j].question[strcspn(exams[examCount].questions[j].question,
"\n")] = '\0';

                for (int k = 0; k < 4; k++) {
                    fgets(exams[examCount].questions[j].options[k], MAX_OPTION_LENGTH,
file);

exams[examCount].questions[j].options[k][strcspn(exams[examCount].questions[j].options[k
], "\n")] = '\0';
                }

                fscanf(file, " %c", &exams[examCount].questions[j].answer);
                fgetc(file); // Remove newline character after answer
            }
            examCount++;
        }
        fclose(file);
    }
}


void loadExams() {
    loadExamsFromText(); // Load from text file
    FILE *file = fopen("exams.dat", "rb");
    if (file) {
        while (fread(&exams[examCount], sizeof(Exam), 1, file)) {
            examCount++;
        }
        fclose(file);
    }
}

void saveExams() {
    saveExamsToText(); // Save to text file
    FILE *file = fopen("exams.dat", "wb");
    if (file) {
        fwrite(exams, sizeof(Exam), examCount, file);
        fclose(file);
    }
}

void createExam() {
    if (examCount >= MAX_EXAMS) {
        printf("Exam limit reached.\n");
        return;
    }

    Exam newExam;
    printf("\nEnter exam name: ");
```

```
scanf("%s", newExam.examName);


printf("\nEnter number of questions: ");

scanf("%d", &newExam.numQuestions);


for (int i = 0; i < newExam.numQuestions; i++) {

    printf("\nEnter question %d: ", i + 1);

    getchar();

    fgets(newExam.questions[i].question, 256, stdin);

    newExam.questions[i].question[strcspn(newExam.questions[i].question, "\n")] = '\0';


    printf("\nEnter options A, B, C, D:\n");

    printf("Option A: ");

    fgets(newExam.questions[i].options[0], MAX_OPTION_LENGTH, stdin);

    newExam.questions[i].options[0][strcspn(newExam.questions[i].options[0], "\n")] = '\0';


    printf("Option B: ");

    fgets(newExam.questions[i].options[1], MAX_OPTION_LENGTH, stdin);

    newExam.questions[i].options[1][strcspn(newExam.questions[i].options[1], "\n")] = '\0';


    printf("Option C: ");

    fgets(newExam.questions[i].options[2], MAX_OPTION_LENGTH, stdin);

    newExam.questions[i].options[2][strcspn(newExam.questions[i].options[2], "\n")] = '\0';
```

```
        printf("Option D: ");
        fgets(newExam.questions[i].options[3], MAX_OPTION_LENGTH, stdin);
        newExam.questions[i].options[3][strcspn(newExam.questions[i].options[3], "\n")] = '\0';

        printf("\nEnter correct answer (A, B, C, D): ");
        scanf(" %c", &newExam.questions[i].answer);
    }

    exams[examCount++] = newExam;
    saveExams();
    printf("\n\t\tExam created successfully.\n");
}

void takeExam() {
    char examName[100];
    printf("\nEnter the exam name you want to take: ");
    scanf("%s", examName);

    for (int i = 0; i < examCount; i++) {
        if (strcmp(exams[i].examName, examName) == 0) {
            int score = 0;
            for (int j = 0; j < exams[i].numQuestions; j++) {
                char userAnswer;
                printf("Q%d: %s\n", j + 1, exams[i].questions[j].question);
                printf("A. %s\n", exams[i].questions[j].options[0]);

                printf("B. %s\n", exams[i].questions[j].options[1]);

                printf("C. %s\n", exams[i].questions[j].options[2]);

                printf("D. %s\n", exams[i].questions[j].options[3]);

                printf("\nYour answer: ");

                scanf(" %c", &userAnswer);


                if (userAnswer == exams[i].questions[j].answer) {

                    score++;

                }

            }

            printf("\nYour score: %d/%d\n", score, exams[i].numQuestions);

            return;

        }

    }


    printf("Exam not found.\n");

}


void adminMenu() {

    int choice;

    do {

        printf("\n\t\tAdmin Menu\n");

        printf("1. Create Exam\n");
        printf("2. Exit\n");
        printf("\n\tEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createExam();
                break;
            case 2:
                printf("\n\t\tExiting admin menu...\n");
                break;
            default:
                printf("Invalid choice.\n");
                break;
        }
    } while (choice != 2);
}

void userMenu() {
    char loggedInUser[100];
    int loggedIn = 0;
    int choice;
```

```
do {
    if (!loggedIn) {
        printf("\n\t\tUser Menu\n");
        printf("1. Register\n");
        printf("2. Login\n");
        printf("3. Exit\n");
        printf("\n\tEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                registerUser();
                break;
            case 2:
                loggedIn = loginUser(loggedInUser);
                break;
            case 3:
                printf("\n\tExiting user menu...\n");
                break;
            default:
                printf("Invalid choice.\n");
                break;
        }
    } else {
        printf("\nUser Logged in as %s\n", loggedInUser);
```

```
            printf("1. Take Exam\n");
            printf("2. Logout\n");
            printf("\n\tEnter your choice: ");
            scanf("%d", &choice);
            switch (choice) {
                case 1:
                    takeExam();
                    break;
                case 2:
                    loggedIn = 0;
                    printf("\n\tLogged out from exam successfully.\n");
                    break;
                default:
                    printf("Invalid choice.\n");
                    break;
            }
        }
    } while (choice != 3);
}
int main() {
    loadUsers();
    loadExams();
    int choice;
    do {
        printf("\n\t\tMain Menu\n");

        printf("1. Admin Login\n");

        printf("2. User Menu\n");

        printf("3. Exit\n");

        printf("\n\tEnter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                adminMenu();

                break;

            case 2:

                userMenu();

                break;

            case 3:

                printf("\n\tExiting the program...\n");

                break;

            default:

                printf("Invalid choice.\n");

                break;

        }

    } while (choice != 3);


    return 0;

}
```