Istanbul Health and Technology University
Faculty of Engineering and Natural Sciences
**Software Engineering Department**

# COURE PROJECT – SWE208
# Computing Systems

## PROJECT TITLE

**Team Members**

Kemal Berfhat Kırklar 220610029
Tuğçe Aycı 220610039
Celal Dinç 220610021

## Project Description

CryptoDiary is a terminal-based diary management application developed using the C programming language. Its primary purpose is to allow users to securely write, store, and manage their personal diary entries. Each diary note is encrypted using a password provided by the user, ensuring that only the rightful owner can access the content. This project was inspired by the need for a lightweight yet effective text encryption and organization system, with the goal of combining fundamental computer systems concepts into a single application.

The intended users of CryptoDiary are individuals who prefer to keep private notes on their computer without using third-party applications. By storing data in a custom binary format and implementing a simple Caesar Cipher encryption, this project also aims to demonstrate critical computing system topics such as file input/output, data representation (ASCII encoding), pointer usage, and memory safety.

CryptoDiary operates entirely within the terminal. Users interact with a clean and user-friendly menu system to create new entries, view existing ones, delete them, or recover forgotten passwords using security questions. The program enforces input validation, password protection, and basic encryption to provide a functional and educational example of applied computing systems principles.

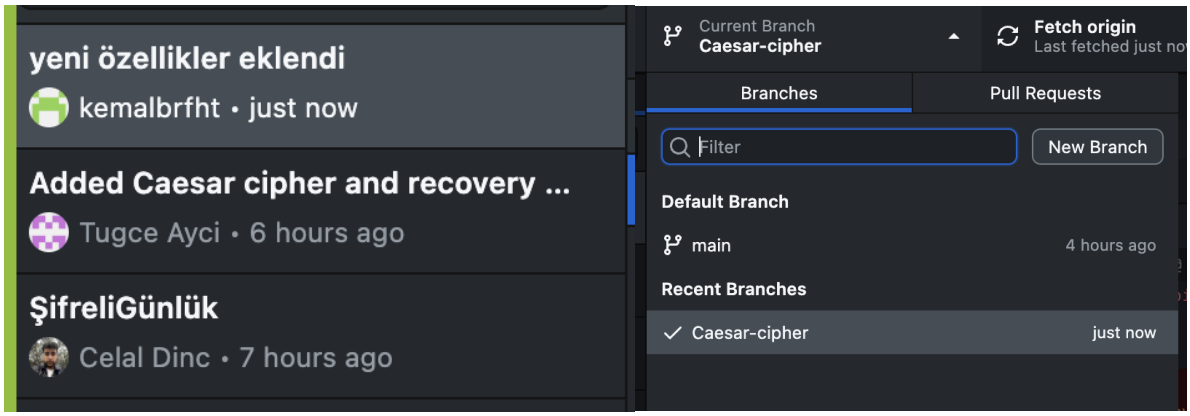
## Key Features

- **Feature 1: Secure Diary Entry Creation** Users can write personal diary notes and save them with a title, content, password, and recovery question. Before saving, the content is encrypted using a Caesar Cipher algorithm, which shifts each character in the message by a fixed key derived from the user-provided password. This ensures that the file's content cannot be read without proper decryption.
- **Feature 2: Password-Protected Viewing** When a user wants to read a saved diary entry, the application prompts for the password. The entered password is compared with the one stored internally (in encrypted form). If the password matches, the content is decrypted and displayed; otherwise, access is denied.
- **Feature 3: Organized Entry Listing and Deletion** The application scans the CryptoDiary folder and lists all diary entries (based on file names). Users can select an entry to delete by confirming their password, providing a clean and organized way to manage old records.
- **Feature 4: Password Recovery System** Each entry includes a recovery question and answer. If a user forgets their password, they can retrieve it by correctly answering the pre-defined question. This adds an extra layer of usability and simulates real-world recovery flows found in software applications.

## Technologies

- **Programming Language:** C
- **Development Environment:** Visual Studio Code with GCC compiler on Ubuntu Linux
- **Standard Libraries Used:**
  - `<stdio.h>` for input/output operations
  - `<stdlib.h>` for memory and utility functions
  - `<string.h>` for string manipulation
  - `<time.h>` for recording timestamps
  - `<sys/stat.h>` for file system operations
  - `<unistd.h>` for POSIX directory access
- **Version Control:** Git was used for source control and all versions of the code were maintained and pushed to a private GitHub repository for transparency and collaboration.

https://github.com/Celaldnc/CryptoDiary.git




## Implementation

The core of CryptoDiary lies in its ability to combine essential computer system concepts into a single, practical program. Below is a detailed breakdown of how the project was implemented:

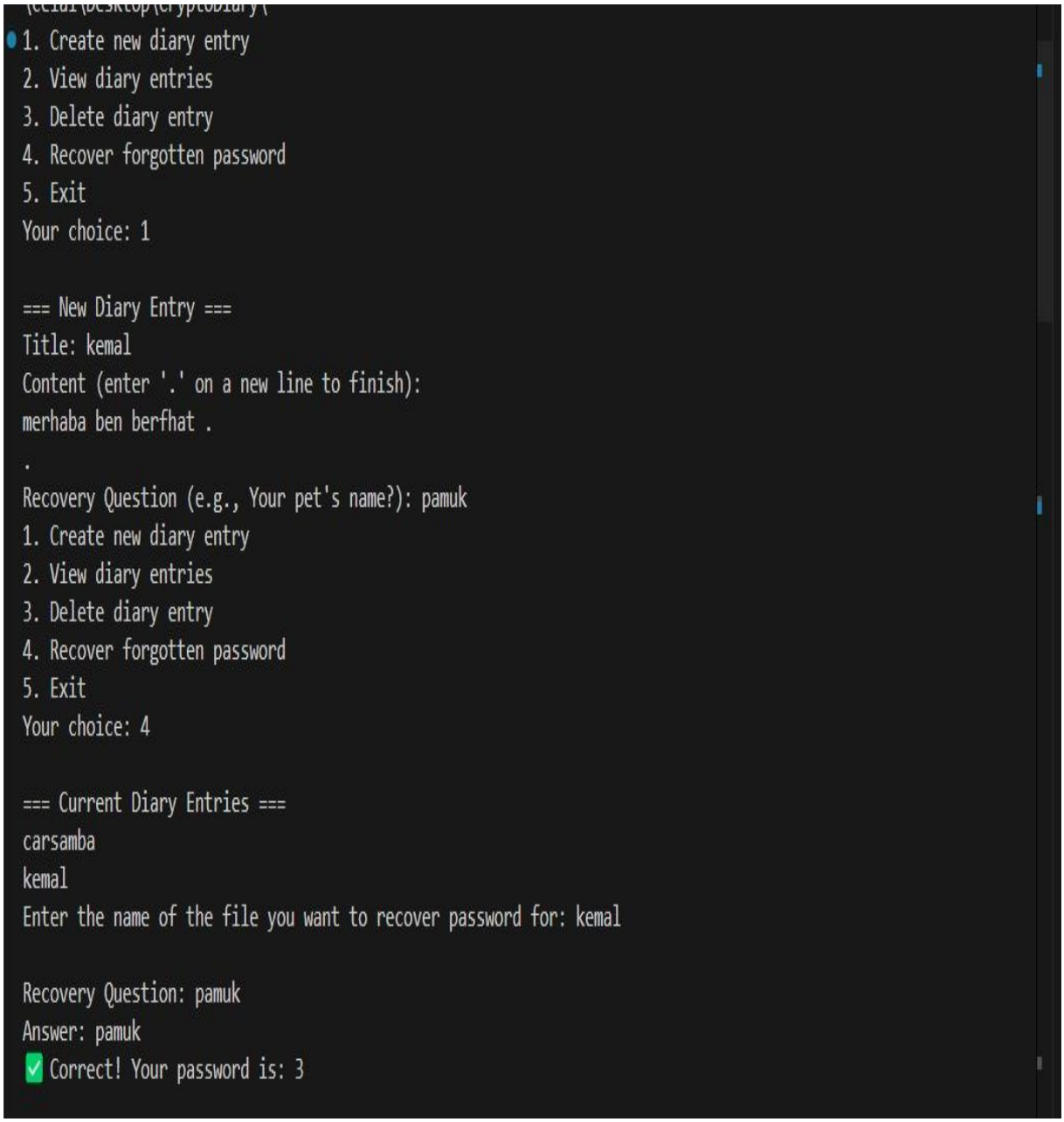
- **Data Structures and Storage** The diary entries are represented using a custom structure `DiaryEntry`, which holds fields such as the title, content, password, date of creation, and recovery question/answer. The structure is written to and read from disk in binary format using `fwrite()` and `fread()`, enabling both space efficiency and direct data representation.

- **Encryption and Decryption** The Caesar Cipher method is used for encrypting the diary content. This algorithm was selected for its simplicity and educational value. Each character in the note is shifted by a numeric key derived from the user's password. While not secure for professional cryptography, it effectively demonstrates the ASCII manipulation and control logic required in systems programming. Decryption reverses this transformation.

```
void caesarEncrypt(char* text, int key) {
    for (int i = 0; text[i] != '\0'; i++) {
        unsigned char c = text[i];
        if (c >= 'a' && c <= 'z') text[i] = ((c - 'a' + key) % 26) + 'a';
        else if (c >= 'A' && c <= 'Z') text[i] = ((c - 'A' + key) % 26) + 'A';
        else if (c >= '0' && c <= '9') text[i] = ((c - '0' + key) % 10) + '0';
        else if (c < 128) text[i] = c;
    }
}
```

- **Input/Output Handling** Files are saved on the user's Desktop inside a folder named "CryptoDiary". The program dynamically constructs file paths and ensures the target directory exists using `mkdir` and `stat`. Entries are saved with `.txt` extension but contain encrypted binary content. When listing entries, `popen()` is used to execute shell commands and retrieve filenames.
- **Password Validation and Recovery** During reading or deletion, the user is prompted to enter the password. This input is validated against the stored value. If the user forgets their password, they can answer a custom recovery question. This simulates real-world authentication fallback systems.
- **Challenges and Solutions** One challenge was ensuring that user input (especially multiline diary content) didn't overflow the internal buffers. This was addressed by using `fgets()` for safe input and applying strict maximum lengths. Another challenge involved storing binary data in text files, which was solved by using fixed-size structs and binary file operations.
- **Performance and Usability Considerations** Although the Caesar Cipher is not computationally intensive, we optimized the input validation to ensure responsiveness. Menus are clearly labeled, and feedback messages guide the user through each step. The system avoids memory leaks by keeping stack-based variables and eliminating the need for manual memory allocation.
- **Extensibility** CryptoDiary was designed with extensibility in mind. Additional features like entry editing, stronger encryption (e.g., XOR with dynamic keys), or syncing entries to the cloud can be added in future versions. The modular design of the codebase supports such expansions.

Çıktı:s


1. Create new diary entry
2. View diary entries
3. Delete diary entry
4. Recover forgotten password
5. Exit
Your choice: 1
=== New Diary Entry ===
Title: kemal
Content (enter '.' on a new line to finish):
merhaba ben berfhat .
.
Recovery Question (e.g., Your pet's name?): pamuk
1. Create new diary entry
2. View diary entries
3. Delete diary entry
4. Recover forgotten password
5. Exit
Your choice: 4
=== Current Diary Entries ===
carsamba
kemal
Enter the name of the file you want to recover password for: kemal
Recovery Question: pamuk
Answer: pamuk
✅ Correct! Your password is: 3