A PROJECT REPORT ON

DIGITAL TIME CAPSULE MANAGEMENT SYSTEM

Submitted by

SIVARAJ T **(113224071091)**

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

VELAMMAL ENGINEERING COLLEGE, CHENNAI-66.
(An Autonomous Institution, Affiliated to Anna University, Chennai)

OCTOBER-2025

VELAMMAL ENGINEERING COLLEGE

CHENNAI-66

BONAFIDE CERTIFICATE

Certified that this project report " **DIGITAL TIME CAPSULE MANAGEMENT SYSTEM**" is the Bonafide work of **SIVARAJ T (113224071091),** who carried out the mini project under my supervision.

SUPERVISOR

Mr Mohan Raj K R

Assistant Professor,

Dept .of Information Technology,

Velammal Engineering College,

Ambattur Red Hills Rd, Surapet,

Chennai-600066.

PROFESSOR AND HEAD

Dr.Jeevaa Katiravan

Head of the Department,

Dept .of Information Technology,

Velammal Engineering College,

Ambattur Red Hills Rd, Surapet,

Chennai-600066.

CERTIFICATE OF EVALUATION

COLLEGE NAME :VELAMMAL ENGINEERING COLLEGE
BRANCH :INFORMATION TECHNOLOGY
SEMESTER :III

COURSE OUTCOME:		Apply the concepts of exeception handling and STL to build related programs	
NO	PROJECT MEMBERS	TITLE OF THE PROJECT	PROJECT SUPERVISOR
	SIVARAJ T	DIGITAL TIME CAPSULE MANAGEMENT SYSTEM	Mr Mohan Raj K R
MARKS OBTAINED			

This report of project work has been submitted by the above student in partial fulfillment for the award of Bachelor of INFORMATION TECHNOLOGY Degree was evaluated and confirmed to be reports of the work done by the above student and then assessed.

Submitted for Internal Evaluation held on.....

Internal Examiner

ABSTRACT

This project presents a console-based Digital Time Capsule System developed in C++, designed to help users securely store personal messages or memories to be accessed at a future date. The system demonstrates key concepts of Object-Oriented Programming (OOP), file handling, encryption, exception handling, and Standard Template Library (STL) usage in a practical and engaging way.

Users can create, view, and manage digital time capsules by providing a username, capsule name, password, and a target year for opening. Capsule names are stored in STL containers such as vector, while messages are encrypted using a simple Caesar cipher to ensure privacy. Files are created dynamically to maintain persistent data across sessions. The project also incorporates input validation, modular functions, and robust error handling to enhance code reusability, maintainability, and user experience.

By combining secure message storage with modern programming techniques, this project provides a strong foundation for beginners to understand real-world C++ applications. It can be further extended to include advanced encryption algorithms, graphical user interfaces (GUI), or cloud-based storage integration for improved functionality, security, and accessibility.

ACKNOWLEDGEMENT

The authors gratefully acknowledge to their beloved CHAIRMAN. Thiru M V MUTHURAMALINGAM, for providing the facilities and constantly encouraging and supporting during the study of course. Sincere and grateful acknowledgement is expressed to the respectable PRINCIPAL, Dr. S SATISH KUMAR, for his encouragement to complete the project. The constant support, guidance and immense encouragement rendered by our HEAD OF THE DEPARTMENT, Dr. JEEVAA KATIRAVAN in completing the project is gratefully acknowledged and placed in record. His inspiration and moral support have been inestimable in increasing their knowledge. Heartfelt sincere thanks and deep gratitude to all faculty members of the Department of INFORMATION TECHNOLOGY, VELAMMAL ENGINEERING COLLEGE for their support on technical background. Our friends and family are also remembered for their immense support and cooperation. In this aspect, we are eternally grateful to all. We express our sincere gratitude to the Project Guide Mr Mohan Raj K R Assistant Professor, Department of Information Technology for their invaluable guidance in shaping of this project without them this project would not have been possible. Our thanks to all other Faculty and Non- Teaching staff members of our department for their support and peers for having stood by me and helped me to complete this project

TABLE OF CONTENTS

INTRODUCTION

The Digital Time Capsule System is a console-based application developed in C++ that allows users to securely store personal messages or memories and retrieve them at a specified future date. The concept of a time capsule — preserving thoughts, emotions, or memories for reflection in the future — is brought to life using programming, file handling, and modern C++ techniques.

This project demonstrates essential Object-Oriented Programming (OOP) principles such as encapsulation, abstraction, and modular design. Users can create, list, and open digital time capsules through a simple and interactive menu-driven interface. Each capsule is protected by a password and stored as an encrypted text file, ensuring privacy until the target year is reached.

The program utilizes file handling for persistent data storage, while STL containers such as vector are used to manage capsule lists dynamically, improving efficiency, scalability, and code clarity. A simple Caesar cipher encryption algorithm safeguards messages from unauthorized access, and exception handling ensures smooth operation by managing invalid inputs, incorrect passwords, and attempts to open capsules before their designated year.

Designed with beginners in mind, the project emphasizes clarity, modularity, and maintainability. It serves as a practical example of how modern C++ programming techniques, including OOP, STL, file handling, encryption, and exception safety, can be applied to real-world problems. Future enhancements may include advanced encryption methods, multi-user management, and graphical interfaces for enhanced usability and accessibility.

PROJECT OUTLINE

The Digital Time Capsule System is a console-based C++ application designed to allow users to securely store personal messages or memories for future retrieval. The project demonstrates the practical implementation of Object-Oriented Programming (OOP) concepts, file handling, data encryption, Standard Template Library (STL) features, and exception management, making it an excellent learning tool for beginner programmers.

The system operates through a menu-driven interface that offers three main functionalities — creating a new capsule, opening an existing capsule, and listing all capsules associated with a user. When creating a capsule, users enter their name, secret message, and password along with the target year for opening. The message is encrypted using a simple Caesar cipher technique to ensure privacy and stored in a uniquely named text file for secure access.

File handling is used to maintain user data persistently, while STL containers such as vector are utilized to manage capsule lists dynamically, ensuring flexibility and scalability. Each user has a dedicated list file that records all their created capsules, enabling easy organization and retrieval. When opening a capsule, the program validates the entered password and ensures that the current year matches or exceeds the target year before decrypting and displaying the stored message.

Exception handling is incorporated to detect and manage errors such as invalid inputs, missing files, incorrect passwords, or attempts to open capsules before their designated year. This enhances the reliability of the application and ensures a smooth and secure user experience.

The project's modular structure, with well-defined functions and encapsulated logic, promotes readability, maintainability, and reusability. It serves as a strong example of how modern C++ programming techniques—including OOP principles, STL usage, and exception safety—can be combined to create a secure, efficient, and user-friendly application. Future enhancements may include stronger encryption algorithms, graphical interfaces, or cloud-based storage integration for improved functionality and accessibility.

TOOLS AND PLATFORM

The Digital Time Capsule System was developed using C++ as the core programming language, utilizing its object-oriented programming (OOP) capabilities, file handling, exception management, and Standard Template Library (STL) features such as vector for efficient and dynamic data management. The project also incorporates a simple Caesar cipher encryption algorithm to ensure the privacy of stored messages.

The development and testing were carried out in Visual Studio Code (VS Code), a lightweight yet powerful source-code editor that supports C++ through extensions like the C/C++ compiler and debugger. VS Code provided an intuitive interface for writing, organizing, and debugging code, making it suitable for both beginners and experienced developers. Its features, such as syntax highlighting, real-time error detection, and integrated terminal commands, enhanced productivity and code quality.

The application was executed on a Windows platform, compiled using the GCC compiler via MinGW or similar toolchains. This environment ensured smooth compilation, execution, and compatibility with standard C++ libraries.

Overall, the combination of C++, STL, file handling, encryption, exception handling, and VS Code as the development platform enabled a structured, reliable, and user-friendly approach to building the Digital Time Capsule System.

ALGORITHM

1. Start the program
2. Initialize variables for username, capsule name, message, password, and target year
3. Display the main menu with options:
    1. Create a New Capsule
    2. Open Existing Capsule
    3. List My Capsules
    4. Exit
4. Repeat until user chooses Exit:
    1. Prompt the user to enter a choice
    2. Read user input
5. If choice is 1 (Create a New Capsule):
    1. Prompt for username
    2. Prompt for capsule name
    3. Prompt for secret message
    4. Prompt to set a password
    5. Prompt for target year
    6. Validate that target year is not in the past
        a. If invalid, display error message
    7. Encrypt the message using Caesar cipher
    8. Save capsule details (created year, target year, password, encrypted message) to a uniquely named file
    9. Add capsule name to user's capsule list file
    10. Display success message

6. If choice is 2 (Open Existing Capsule):
    1. Prompt for username
    2. Prompt for capsule name
    3. Open the corresponding capsule file
        a. If file does not exist, display error message
    4. Read created year, target year, password, and encrypted message from file
    5. Prompt for password
        a. If password is incorrect, display error message
    6. Check if current year is greater than or equal to target year
        a. If not, display message that capsule is still sealed
    7. Decrypt message and display it
7. If choice is 3 (List My Capsules):
    1. Prompt for username
    2. Open user's capsule list file
    3. If file does not exist, display message that no capsules are found
    4. Read and display all capsule names stored in the list
8. If choice is 4 (Exit):
    1. Display exit message and terminate program
9. End program

SOURCE CODE

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctime>
#include <stdexcept>
#include <vector>
#include <algorithm>
using namespace std;

class TimeCapsule {
private:
    string username, capsuleName, message, password;
    int targetYear;

    // Simple Caesar cipher for encryption/decryption
    string encrypt(const string &text) {
        string result = text;
        transform(result.begin(), result.end(), result.begin(),
                []char c) { return c + 3; });
        return result;
    }

    string decrypt(const string &text) {
        string result = text;
        transform(result.begin(), result.end(), result.begin(),
                []char c) { return c - 3; });
        return result;
    }

public:
    void createCapsule() {
        cout << "\nEnter your username: ";
        getline(cin, username);
        cout << "Enter a name for this capsule: ";
        getline(cin, capsuleName);
        cout << "Enter your secret message: ";
        getline(cin, message);
        cout << "Set a password: ";
        getline(cin, password);
        cout << "Enter open year: ";
        cin >> targetYear;
        cin.ignore();
```

```
time_t t = time(0);
    tm *now = localtime(&t);
    int currentYear = now->tm_year + 1900;

    try {
      if (targetYear < currentYear)
        throw invalid_argument("Error: Target year cannot be in the past!");

      // Maintain capsule list using STL vector
      vector<string> capsuleList;

      // Read existing list (if any)
      ifstream readList(username + "_capsules.txt");
      string capsule;
      while (getline(readList, capsule)) {
        capsuleList.push_back(capsule);
      }
      readList.close();

      // Add new capsule to list and write back
      capsuleList.push_back(capsuleName);
      ofstream writeList(username + "_capsules.txt");
      for (const auto &c : capsuleList)
        writeList << c << endl;
      writeList.close();

      // Save capsule data
      ofstream file(username + "_" + capsuleName + ".txt");
      if (!file)
        throw runtime_error("Error: Unable to create capsule file!");

      file << currentYear << endl;
      file << targetYear << endl;
      file << password << endl;
      file << encrypt(message) << endl;
      file.close();

      cout << "\nCapsule '" << capsuleName << "' created successfully!\n";
    } catch (exception &e) {
      cout << e.what() << endl;
    }
  }
```

```
void openCapsule() {
    cout << "\nEnter your username: ";
    getline(cin, username);
    cout << "Enter capsule name: ";
    getline(cin, capsuleName);

    string filename = username + "_" + capsuleName + ".txt";
    ifstream file(filename);

    try {
      if (!file)
        throw runtime_error("Error: Capsule not found!");

      int createdYear;
      string storedPassword, encryptedMsg;
      file >> createdYear >> targetYear;
      file.ignore();
      getline(file, storedPassword);
      getline(file, encryptedMsg);
      file.close();

      cout << "Enter your password: ";
      string inputPassword;
      getline(cin, inputPassword);

      if (inputPassword != storedPassword)
        throw invalid_argument("Error: Incorrect password!");

      time_t t = time(0);
      tm *now = localtime(&t);
      int currentYear = now->tm_year + 1900;

      if (currentYear < targetYear)
        throw logic_error("Capsule is still sealed! Come back in the future!");

      cout << "\nOpening Capsule '" << capsuleName << "'...\n";
      cout << "Created in: " << createdYear << " | Target Year: " << targetYear << "\n";
      cout << "Your Message:\n" << decrypt(encryptedMsg) << endl;

    } catch (exception &e) {
      cout << e.what() << endl;
    }
  }
```

```
void listCapsules() {
        cout << "\nEnter your username: ";
        getline(cin, username);
        ifstream listFile(username + "_capsules.txt");

        try {
            if (!listFile)
                throw runtime_error("No capsules found for this user!");

            vector<string> capsules;
            string capsule;
            while (getline(listFile, capsule)) {
                capsules.push_back(capsule);
            }
            listFile.close();

            if (capsules.empty())
                cout << "(No capsules created yet)\n";
            else {
                cout << "\nAvailable Capsules for " << username << ":\n";
                for (const auto &c : capsules)
                    cout << "- " << c << endl;
            }
        } catch (exception &e) {
            cout << e.what() << endl;
        }
    }
};
```

```
int main() {
    TimeCapsule capsule;
    int choice;

    do {
        cout << "\n===== DIGITAL TIME CAPSULE =====\n";
        cout << "1. Create a New Capsule\n";
        cout << "2. Open Existing Capsule\n";
        cout << "3. List My Capsules\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore();

        switch (choice) {
        case 1:
            capsule.createCapsule();
            break;
        case 2:
            capsule.openCapsule();
            break;
        case 3:
            capsule.listCapsules();
            break;
        case 4:
            cout << "Goodbye! Your memories are safe.\n";
            break;
        default:
            cout << "Invalid choice! Try again.\n";
        }
    } while (choice != 4);

    return 0;
}
```

OUTPUT


===== DIGITAL TIME CAPSULE =====
1. Create a New Capsule
2. Open Existing Capsule
3. List My Capsules
4. Exit
Enter your choice: 1
Enter your username: sivaraj
Enter a name for this capsule: sih
Enter your secret message: i an sivaraj. i have selected in sih.
Set a password: Siva@9677
Enter open year: 2025
Capsule 'sih' created successfully!
===== DIGITAL TIME CAPSULE =====
1. Create a New Capsule
2. Open Existing Capsule
3. List My Capsules
4. Exit
Enter your choice: 1
Enter your username: sivaraj
Enter a name for this capsule: symposium
Enter your secret message: i have participated in SRM vadaplani branch symposium
.
Set a password: Siva@9677
Enter open year: 2026
Capsule 'symposium' created successfully!
===== DIGITAL TIME CAPSULE =====
1. Create a New Capsule
2. Open Existing Capsule
3. List My Capsules
4. Exit
Enter your choice: 2
Enter your username: sivaraj
Enter capsule name: sih
Enter your password: Siva@9677
Opening Capsule 'sih'...
Created in: 2025 | Target Year: 2025
Your Message:
i an sivaraj. i have selected in sih.
===== DIGITAL TIME CAPSULE =====
1. Create a New Capsule
2. Open Existing Capsule
3. List My Capsules
4. Exit
Enter your choice: 2
Enter your username: sivaraj
Enter capsule name: symposium
Enter your password: Siva@9677
Capsule is still sealed! Come back in the future!
===== DIGITAL TIME CAPSULE =====
1. Create a New Capsule
2. Open Existing Capsule
3. List My Capsules
4. Exit
Enter your choice: 3
Enter your username: sivaraj
Available Capsules for sivaraj:
- sih
- symposium

FUTURE ENHANCEMENTS

The Digital Time Capsule System can be further improved and extended with the following enhancements:

1. Advanced Encryption Methods:
   - Replace the simple Caesar cipher with stronger encryption algorithms such as AES or RSA to enhance the security of stored messages.
2. Graphical User Interface (GUI):
   - Implement a user-friendly GUI using frameworks like Qt or SFML to make the system more interactive and visually appealing.
3. Multi-User Management:
   - Allow multiple users to create and manage capsules simultaneously, with separate accounts and secure login credentials.
4. Cloud-Based Storage Integration:
   - Enable storing capsules in cloud storage or a database for remote access and better data persistence.
5. Notification System:
   - Add reminders or notifications to alert users when a capsule's target year has been reached.
6. Search and Sort Features:
   - Allow users to search capsules by name or date and sort them for easier management.
7. Backup and Restore Functionality:
   - Implement automatic backup of capsules to prevent data loss and allow easy restoration of saved messages.
8. Mobile or Web Version:
   - Extend the system to mobile or web platforms to increase accessibility and usability.

DRAWBACKS

While the Digital Time Capsule System demonstrates key programming concepts such as OOP, STL usage, file handling, encryption, and exception management, it has several limitations that may affect its practical use:

1. Single-User Focus:
    - The system is designed primarily for individual users. It does not support multiple users accessing and managing capsules simultaneously.
2. Basic Security:
    - Although messages are encrypted using a simple Caesar cipher and protected with a password, the encryption is not strong enough for highly sensitive information.
3. Console-Based Interface:
    - The menu-driven console interface, while simple and easy to use, lacks graphical elements, which may reduce user engagement and accessibility for non-technical users.
4. Limited Error Handling:
    - While basic exceptions are handled, the system does not cover all possible edge cases, such as file corruption or unexpected input formats.
5. No Advanced Search or Sorting:
    - Users cannot search for capsules by keywords, dates, or other criteria, making navigation through multiple capsules cumbersome.
6. No Backup or Cloud Storage:
    - Capsule data is stored locally in text files, meaning that data can be lost if files are deleted or the system crashes. There is no built-in backup or cloud synchronization feature.
7. Lack of Notifications or Alerts:
    - The system does not notify users when a capsule reaches its target year, requiring them to check manually.

CONCLUSION

The Digital Time Capsule System project successfully demonstrates the practical application of core programming concepts using C++, including Object-Oriented Programming (OOP), file handling, encryption, exception management, and Standard Template Library (STL) usage. Through its structured design and interactive menu-driven interface, the system allows users to create, open, and manage digital time capsules securely while handling errors gracefully.

By incorporating modular functions, encapsulated logic, STL containers like vector, and basic encryption, the project promotes clean, maintainable, and scalable code. The use of Visual Studio Code as the development platform further enhanced the coding and debugging experience, making it accessible and manageable for beginners.

While the current system is limited to a console interface, single-user focus, and basic encryption, it provides a strong foundation for future enhancements such as advanced encryption methods, graphical user interfaces (GUI), multi-user management, cloud storage integration, and notifications.

Overall, this project serves as an effective learning tool for understanding how structured programming, file management, and error handling can be combined to solve real-world problems. It also lays the groundwork for more advanced software development opportunities in secure data storage and personal information management.

REFERENCES

1. The C++ Programming Language, 4th Edition, B. Stroutstrup, Pearson Education, 2013. 4. Programming with C++, 7th edition, E.Balagurusamy, Tata McGraw Hill, 2017. 5. Computing fundamentals with C++, Object Oriented Programming & design", Rick Mercer, "Mac Millan Press IInd edition 1995

TEXT BOOKS

1. The Complete Reference C++, 5th Edition, Herbert Scheldt, Tata McGraw Hill, 2012. 2. Problem solving with C++: The Object of Programming, 10th Edition, Walter Savitch, Pearson Education. 2018.