CPE 112 Computer Programming

Pool Villa Booking System

Members

Kawinpob	Amnuaywittayakul	67070503403
Pawarut	Kumnungwut	67070503422
Phureeruch	Satithangkul	67070503471

Table of Contents

Abstract

This project presents a terminal-based Pool Villa Booking System developed in the C programming language. The system is designed to serve two types of users: customers and property managers. Customers can browse and book villas, view detailed property information, manage favorites, and cancel bookings with an automatic refund policy. Managers are provided with tools to add, edit, and remove villa listings, as well as to manage date-specific availability through an interactive calendar interface. Data persistence is handled through CSV files, ensuring ease of storage, portability, and simplicity in editing. The system includes robust date validation and conflict checking to prevent overlapping bookings and maintain accurate records. By leveraging structured data, user role separation, and clear console-based interactions, the system aims to streamline the pool villa rental process while maintaining reliability, usability, and scalability for future extensions.

1. Introduction

This system is designed to simplify the process of booking pool villas for customers and streamline villa management tasks for property managers. Customers can browse available villas, view detailed information, make bookings, manage favorites, and cancel reservations with automatic refunds. Managers are provided with tools to add, edit, or delete villa listings, control calendar availability, and review booking history. The system uses a terminal-based interface developed in the C programming language, with all data stored in CSV files for easy management and portability. Calendar-based availability and booking validation help prevent double-bookings and ensure an accurate schedule. The goal is to provide a complete, efficient booking experience with a user-friendly interface and a reliable back-end.

2. Objective

The objective of this project is to develop a comprehensive booking and management system for pool villas, enabling both customers and property managers to efficiently interact with the system. For managers, the system offers features such as viewing and managing villa listings, performing CRUD operations on property data, setting calendar availability, blocking dates for maintenance, and reviewing booking and cancellation histories. Customers can browse available villas, view detailed information, add favorites, make bookings for specific dates, and cancel bookings with an automatic refund policy. The system also includes a calendar-based availability interface to prevent overlapping bookings and ensure smooth scheduling. All data is stored and updated using CSV files, ensuring easy management and future scalability of the platform.

3. Solution with Functionalities

This Pool Villa Booking System addresses the challenges of managing short-term villa rentals through a clear separation of roles and features. It provides a robust console-based interface backed by structured CSV data storage and dynamic memory management using C.

Customer Side Features:

- **Browse Villas:** View a list of all available villas with basic details like price, location, and rating.
- **View Details:** Access full information such as address, facilities, transportation, and nearby landmarks for any villa.
- **Book a Villa:** Select check-in and check-out dates. The system validates availability using Calendar.csv and confirms booking.
- **Cancel Booking:** Customers can cancel bookings and receive a 90% refund. Cancellations are logged in Cancelled_bookings.csv.
- **Manage Favorites:** Add or remove favorite villas to favorites.csv and view the list later.
- **Calendar Availability View:** A monthly calendar grid highlights available, booked, and blocked days visually for easy booking decisions.

Manager Side Features:

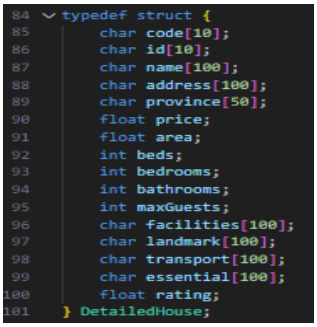
- **Add/Edit/Delete Villas:** Manage property records using Briefly_Info.csv and Detail.csv with full CRUD functionality.
- **Set Calendar Availability:** Use a calendar UI to block/unblock dates or mark them for maintenance in Calendar.csv.
- **Booking History:** Access full logs of past bookings from Booking_history.csv.
- **Override Calendar:** Managers can override availability for urgent maintenance or special bookings.

4. Code Walkthrough with Data Structures

This project is built in the C programming language using a modular design. It utilizes structured data (struct), hash table, binary search tree, 2D arrays (for calendar rendering), and CSV file I/O for persistent storage. The system separates features between customers and managers, each backed by clearly defined data structures.

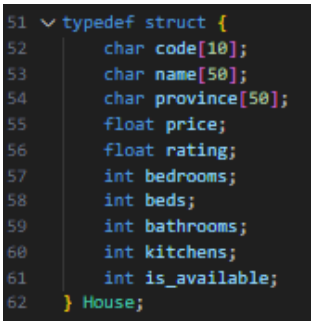
1. Struct Definitions

Stores full villa information loaded from Detail.csv.



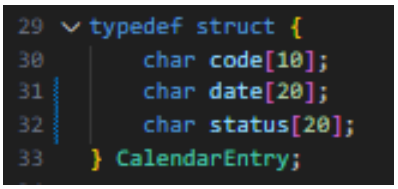
```
typedef struct {
    char code[10];
    char id[10];
    char name[100];
    char address[100];
    char province[50];
    float price;
    float area;
    int beds;
    int bedrooms;
    int bathrooms;
    int maxGuests;
    char facilities[100];
    char landmark[100];
    char transport[100];
    char essential[100];
    float rating;
} DetailedHouse;
```

Used to store and display summarized villa data from Briefly_Info.csv.



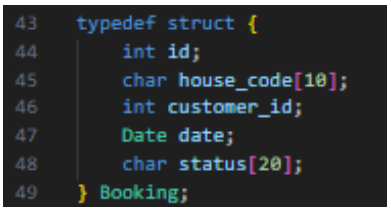
```
typedef struct {
    char code[10];
    char name[50];
    char province[50];
    float price;
    float rating;
    int bedrooms;
    int beds;
    int bathrooms;
    int kitchens;
    int is_available;
} House;
```

Represents date-specific availability loaded from Calendar.csv.



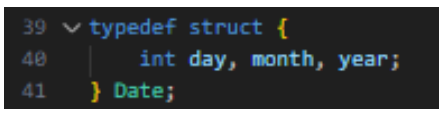
```
typedef struct {
    char code[10];
    char date[20];
    char status[20];
} CalendarEntry;
```

Stores a customer's booking information, written to Booking_history.csv.



```
typedef struct {
    int id;
    char house_code[10];
    int customer_id;
    Date date;
    char status[20];
} Booking;
```

Used internally for booking and availability logic, easier for date comparisons.



```
typedef struct {
    int day, month, year;
} Date;
```

2. Binary Search Tree (BST)

Reason why BST

- Enables efficient in-order traversal to display houses sorted by price
- Scalable: can grow dynamically without needing to reallocate memory
- Faster search and ordering compared to arrays for sorted output

This allows the system to:

• Insert houses based on price during loading
• Display filtered results in ascending order
• Avoid repeated sorting when re-displaying price-based search results

3. Hash Table

Hash tables are used to manage customer favorites efficiently:
• favoriteTable[] stores codes of favorite houses for quick lookup

Reason why Hash Table

• Provides average-case **O(1)** time for insert, delete, and search
• Minimizes the delay in checking if a house is already favorited
• Supports dynamic number of favorites without performance degradation

This allows the system to:

• Mark a house as favorite instantly
• Check for duplicates before saving a favorite
• Remove a favorite without scanning the entire list

4. Calendar Grid (2D Array)

- Renders the availability of each villa per month
- Visually displays: [A] = Available, [B] = Booked, [X] = Blocked

Visual terminal-based monthly calendar UI helps customers/managers see availability status day-by-day.

5. CSV File Integration

All persistent data is stored and read using .csv files. The format allows easy portability, editing, and long-term data handling.

- **Briefly_Info.csv, Detail.csv** – Store summarized and detailed house records respectively.

    - Briefly_Info.csv: Used for list view and searching.
    - Detail.csv: Used for full property details on the booking page.

- **Calendar.csv** – Tracks **daily availability** of each house, including status: Available, Booked, or Blocked.
- **Booking_history.csv, Cancelled_bookings.csv** – Log all **confirmed bookings** and **cancellations with refund** for traceability.
- **favorites.csv** – Stores a list of **customer-selected favorite villas**, referenced by house codes.

5. Time Complexity Analysis

Operation	Time Complexity (With Data Structure)	Time Complexity (Without Data Structure)	Time Improvement
Add Favorite House	O(1) using Hash Table	O(n) searching & appending into file	**Faster**
Remove Favorite House	O(1) hash lookup + O(n) file rewrite	O(n) linear search & rewrite	**Faster**
Check if House is Favorite	O(1) using Hash Table	O(n) scanning file for code	**Faster**
Filter & View Available Houses	O(n) scanning array + optional BST traversal	O(n) same	**Same**
Display Houses Sorted by Price	O(n log n) insertion, O(n) inorder traversal	O(n log n) sort manually before display	**Same**
Search House by Code	O(1) using Hash Table (if applied)	O(n) array linear search	**Faster** (optional)
Insert House to BST (by Price)	O(log n) average, O(n) worst (unbalanced BST)	O(n log n) for manual sort	**Faster**
Booking Insert to CSV	O(1) append to file	O(1) append to file	**Same**

Sync Availability from Calendar	O(n×m) matching house×calendar	O(n×m) same	**Same**
Load Favorite Houses	O(n) file load + O(1) insert each to hash	O(n) file load & later O(n) per search	**Faster**

This system relies on file-based storage (CSV files) and dynamic in-memory structures like linked lists and arrays. The performance of each operation depends on how many records must be processed.

- **Search villa by code/name**: Performed linearly through a list or CSV file. Best case is O(1) if the villa is found early. Worst case O(n) when it's the last entry or not found.
- **Add/Edit/Delete villa**: In a linked list, adding a new villa at the head is O(1). However, editing or deleting requires traversal to find the target node, leading to O(n) in the worst case.
- **Booking conflict check**: When a user selects check-in and check-out dates, the system scans Calendar.csv for any overlap. Checking just a few days (e.g., a weekend stay) is fast—O(1) to O(3)—but longer date ranges increase complexity linearly with the number of days, hence O(d).
- **Cancel booking**: Searches for a matching record in the booking history file. If the record is early, it's fast (O(1)), but if it's deep in the file, it may take O(n) time.
- **Load/display calendar**: Rendering the calendar for a month always processes 28 to 31 days, which is a small, fixed size. Thus, this operation is effectively O(1) in practice and at worst O(31).
- **Favorites management**: Add/remove favorites works with a small list stored in a flat CSV file. It requires a linear search (O(n)) to prevent duplicates or remove a match.
- **Read/Write to CSV**: Every read or write operation processes full files line by line. Since there's no database or indexing, it's always O(n) based on the number of lines in the file.

6. Challenges and Solutions

Data Consistency Across Multiple Files
In some cases, house or calendar data is saved across multiple CSV files such as Briefly_Info.csv, Detail.csv, and Calendar.csv. This creates confusion when one item is edited, as the same house must be updated in all files. The solution to this is to create a logic that synchronizes changes across files using temporary storage and rewriting the files together.

Booking Conflict and Calendar Overlap
When a customer tries to book a house, the system must make sure that the dates are not already booked or blocked by the manager. The problem is checking overlapping dates from the calendar file. The solution to this is to use a linked list to load all the dates and check every entry during booking.

Displaying a Visual Calendar in Terminal
Creating a monthly calendar view in the terminal that matches real-world layout (starting on the right weekday) is difficult. The solution to this is using the built-in time functions and creating a 2D array to print the dates correctly in a grid format.

Difference in Variables due to Miscommunication
When coding separately, a problem occurred where the variables used in different functions did not match. This caused compile-time errors and wrong data assignments. The solution to this is to communicate clearly and unify all struct definitions and variable names used by all members.

Editing Two of the Same Item in Multiple CSV Files
Sometimes a house name or ID appears in more than one CSV file. When one is edited, the other still has the old value, which causes inconsistency. The solution to this issue is creating a temporary variable or file to remember the original name and use it to update all other CSV files.

Managing Favorites and Booking History Without a Database
Since this project uses only CSV files and no database, keeping track of customer favorites and booking history is hard. The solution to this is using linked lists to load data into memory, then writing back to CSV after any changes.

7. Conclusion

The Pool Villa Booking System was successfully developed using C, with features supporting both customers and managers. Key functions like booking, cancellation, calendar availability, and house management were implemented with structured logic and CSV file handling. Challenges such as keeping data consistent across files and handling input validation were solved with proper planning and clear communication. Overall, the system meets its goals and demonstrates practical file-based data management and user interaction in C.

8. Team Member Responsibilities

Name	Student ID	Role & Responsibility
Pawarut Kumnungwut	67070503422	Responsible for all **Manager-side functionalities**, including villa CRUD operations and calendar availability setup.
Kawinpob Amnuaywittayakul	67070503403	Developed **Customer-side features**, including villa browsing, booking, favorites management, and cancellation system.
Phureeruch Satithangkul	67070503471	Overseeing the overall system design, workflow coordination, and implementing all **CSV file handling and integration**.