Coursework 2: Dentist Booking System, Documentation Group Project

2023-24 CST2550 Software Engineering Management and Development

Coursework 2: Dentist Booking System

Module Teacher:
Ahmed Eissa - Module leader (Hendon),
Adam Philpot (Hendon)

Group name: Hyper-Sonic solutions.

Group participants:

- Adam P Solomons - M00899997 Role: Scrum Master, Developer and Tester
- Arbaaz Hussain - M00872279 Role: Developer & Tester
- Novuyo Shumba - M00737611 Role: Secretary
- Simon Sokolowski - M00948848 Role: Developer

Submission date: 21/04/24, 17:30pm

Hyper-Sonic solutions.

Introduction

Project Idea: A dentist booking system for Doctors, Admin, and customer to use.
Our project is designed with the intent help with the daily organisation of appointments and room booking for a dentist clinic. The system handles patient and dentist records and organizing room assignments. It offers an extensive user interface through which users can interact with the system to perform a range of tasks.

Design Decisions

Justification:

Chosen Algorithm: Quick sort.

Quick sort is a sorting algorithm based in the Divide and conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in a stored array. It will repeat this step until the array is sorted. This algorithm has a sharper edge against the others because it has an average case time complexity with versatile applications in various fields.

Unlike merge sort it doesn't require a temporary array to merge the sorted arrays which gives it the advantage of data space. Because of how fast and efficiently it performs in comparison to selection sort, merge sort and insertion sort it has gained a lot of popularity and is usually used for bigger data.

Quick sort has a space complexity of O(logn), making it an excellent choice for situations when space is limited. However, it is considered as an unstable algorithm since it doesn't maintain the key-value pairs initial order and can be difficult to implement if recursion isn't available as it is a recursive process.

Vectors are used to store elements of similar data types, and the size of vectors can grow greatly, for such reasons we believed this would be the best data structure for our program as it allowed for the data stored to expand with the likely addition of new patients, appointment bookings and so on. Vectors have methods like remove /add elements, change elements and access elements. All useful methods that have contributed to the seamless design of our system. Other data structures were considered when drafting the design of our system but in comparison to other data structures like arrays, linked list, graphs, BST and more, vectors proved to be the most efficient and flexible of the list. Allowing us to even implement a variety of other data structures if we deemed it appropriate.

Main components existing in the program:

1. **Data Management (Data Class)**
- Manages all data-related operations, serving as the central hub for access and modifications to the clinic's data, including patients, dentists, appointments, and rooms. It supports dynamic operations to the objects it handles and provides a method to handle saving data to CSV files.

2. **User Interface Management (Interface Class)**
- Handles user interactions, providing a structured way to navigate through the system's features, including viewing records, booking appointments, and adding new entries to the csv. Error handling for user inputs has been implemented to ensure smooth operation.

3. **Patient and Dentist Management**
- **PatientManager and DentistManager**: These classes handle specific operations related to patients and dentists, such as prompting the user for details, creating new objects, and updating the csv.

4. **Appointment Booking and Management (Book Class)**
- Enables the booking of new appointments and management of existing ones. It includes features for booking follow-up appointments, which are linked to original appointments for better tracking. These are communicated to the user when viewing the data.

5. **Display and Reporting (View Class)**
- Responsible for all outputs related to data visualization, such as listing available appointments, all appointments, available rooms, and specific listings based on dentist or patient. Appointments viewed are sorted using the **Quicksort** algorithm.

6. **Calendar Management (Calendar Class)**
- Handles date and time calculations, supporting features like determining dates for follow-up appointments and providing time-related data across the dentist system.

The functional capabilities of our dentist booking system include user interaction; data handling; appointment scheduling and report and visualization. Through a series of menus and prompts, users can interact dynamically with the system to perform tasks like booking appointments, adding new patient or dentist profiles, and viewing detailed reports of appointments. This builds on the usability of the system we found this important to include so our system could be easily integrated into the daily management of dentistry practice.

With data handling we used a robust management of data with capabilities to add, update, and retrieve information efficiently through the use of vectors.
The system is capable of handling direct appointment bookings as well as follow up appointments for regular customers. This is crucial for managing patient care on a long-term basis. Such level of organisation adds to the quality-of-service customers receive.
A comprehensive view of all data in correlation to appointments, room availability and listings tailored to user queries is available through our visualisation feature.
Having our functions existing in their files has made it easier to interact with them as there are multiple programmers on the team and for aesthetic purposes this was the cleanest way to display all the functions our system has.

Testing Methods:

The Testing Methods was used to guarantee the consistency of the reliability and functionality throughout the Dentist Booking System, in the project our teams have implemented various testing methods covering different aspects of the system this includes the following:

1. **Unit Testing:** We conducted unit tests for the individual components of the Dentist Booking System, hence comprising of the following classes such as Dentist, Patient, Room, Appointment, Data, Interface, Book, View, and Calendar. Each of the individual unit test are focused on confirming that it has been implemented correctly for the methods and functionalities within these classes.

2. **Integration Testing:** Integration testing was achieved to estimate the interaction and integration of different modules within the Dentist Booking System. Thus, this concluded Involving the testing in the flow of data and operations between modules such as the following: data management, user interface, appointment booking, and reporting.

3. **User Acceptance Testing (UAT):** UAT was conducted to ensure that the system meets the requirements and expectations of end-users, including the following: doctors, administrators, and customers. We invited stakeholders and third-party testers to interact with the Dentist Booking System and provide their feedback on its usability, functionality, and overall satisfaction.

4. **Boundary Testing:** Our teams has tested the system's behaviour at the boundaries of input ranges to confirm that it handles any extreme values and edge cases appropriately. Furthermore, this included testing for boundary conditions which is related to the following: appointment IDs, patient IDs, and dentist salaries.

5. **Load Testing:** The load testing was performed to assess the Dentist Booking System's performance under usual circumstances and peak load circumstances that can occur. We simulated a high volume of concurrent users accessing the system simultaneously to identify any further performance bottlenecks and confirm that the system remains responsive and stable at all stages throughout the system.

6. **Error Handling Testing:** As a team we deliberately introduced errors and invalid inputs into the system to assess its error handling capabilities. This included testing how the Dentist Booking System responds to incorrect user inputs, database errors, and unexpected exceptions, ensuring that appropriate error messages are being displayed, and the system appropriately handles exceptions that might occur.

7. **Regression Testing:** Whensoever modifications or enhancements were made to the system, regression testing was conducted to ensure that existing functionalities were not adversely affected. This involved re-running previously executed tests to validate that no unintended side effects occurred due to the changes.

8. **Compatibility Testing:** We tested the Dentist Booking System across different platforms hardware's and operating systems to ensure compatibility and consistent behaviour across various environments. This included testing on different versions of computer systems and operating systems commonly used by our target users.

By utilising these testing methods, we have aimed to identify and address any problems or deficiencies in the Dentist Booking System, thus ensuring that it meets the highest standards of quality, reliability, and user satisfaction.

Conclusion

In developing the dental clinic management system, we aimed to integrate sophisticated software engineering practices to enhance the efficiency and reliability of key operational processes within a dental clinic. The system's architecture was designed around solid design principles and well-established patterns, which we selected to ensure scalability and ease of future maintenance. Through this project, we have applied theoretical knowledge in a practical context, demonstrating how strategic design decisions can substantially improve the robustness and functionality of an application.

Limitations and Critical Reflection:

The limitations we experienced as a group was mainly meeting schedules, even though the scrum master and sectary organised them daily meetings became unattainable. Alternatively, as a team we agreed that on some days two team meetings would have to be had to ensure each member was informed of any changes decided and the progress on each section of the project.

Outside of experience group limitations individuals in the group did experience their own problems within their roles such as: The way in which we combatted this was through assisting each other in the tasks we had to complete. Every member took the chanced and stepped out of their role responsibilities to help each other complete the task in time for our deadline. Our daily meeting allowed for this to be possible as we got the chance to discuss current challenges and possible solutions for each challenge.

The development of our dentist booking system, while successful in many aspects, also revealed several limitations that warrant discussion.

One significant limitation is the lack of functionality for adding new dentists and rooms directly within the program interface. Currently, users must manually edit the CSV file containing dentist and room data to make additions or modifications, requiring additional configuration steps beyond the program's runtime. This dependence on manual editing detracts from the system's usability and introduces potential sources of error.

Another limitation is the program's lack of optimization for handling massive datasets. While this may not be a pressing concern for individual clinics with relatively modest data volumes, it could pose scalability challenges in scenarios where the system needs to accommodate larger datasets or support multiple concurrent users.

Additionally, the absence of a graphical user interface (GUI) presents a usability limitation. Without a GUI, the system relies solely on command-line interactions, which may not be intuitive for all users. This limitation restricts the accessibility of the system, particularly for individuals less familiar with command-line interfaces.

Recommendations for Future Improvement:

Reflecting on these limitations, several recommendations emerge to enhance the functionality, usability, and scalability of the dentist booking system in future iterations.

Firstly, implementing features within the program interface to facilitate the addition and management of dentists and rooms would streamline the user experience and reduce reliance on manual CSV editing. Integrating interactive forms or prompts for data entry could simplify the process and minimize errors.

To address scalability concerns, future iterations of the system could incorporate optimizations for handling larger datasets, such as implementing more efficient data structures or database management techniques. This would ensure that the system remains responsive and stable even as data volumes grow.

Finally, the development of a graphical user interface (GUI) would greatly improve the system's usability and accessibility. A GUI could provide intuitive controls and visual feedback, making it easier for users to interact with the system and reducing the learning curve for new users.

By addressing these recommendations in future iterations, we can enhance the overall functionality and usability of the dentist booking system, improving its effectiveness in supporting daily operations within dental clinics.

References

Author:	**Title, Date accessed:**
Programmiz	C++ Vectors [20.3.24]
Harshil Patel	Built In | Quicksort Algorithm: An overview [24.2.24]
Cay Horstmann & Timothy Budd	Big C++ 2nd edition [24.02.24 – 12.04.24]
M. H. Alsuwaiyel	Algorithms Design Techniques and Analysis, Revised edition [24.02.24 – 12.04.24]