POE

PROG7312

Maximilian Walsh – ST10203070

2024

Table of Contents

[Part 3 – Municipal Services Application for South Africa (Project Completion Report and Technology Recommendations) 2](#_Toc181705404)

[Project Completion Report 2](#_Toc181705405)

[Technology Recommendations 4](#_Toc181705406)

[ChatGPT Input/Output 6](#_Toc181705407)

[References 7](#_Toc181705408)

# Part 3 – Municipal Services Application for South Africa (Project Completion Report and Technology Recommendations)

\*ChatGPT was used in my work. See [Chat input/output](#_Chat_Input/Output).\*

## Project Completion Report

#### Overview of the Project

This municipal services application was developed to streamline the reporting and tracking of municipal services requests. Inclusive of the management of service statuses, issue reporting, and local event management. The application comprised key functionalities such as reporting issues, service request status tracking, and event recommendations. Task 3 involved the integration of multiple advanced data structures and algorithms to optimise the application’s performance. Specifically, basic trees, binary trees, binary search trees, AVL trees, red-black trees, heaps, graphs, and minimum spanning trees. Each required structure was implemented to improve the efficiency and scalability of the Service Request Status feature.

#### Challenges Faced in Task 3

The most significant challenge I faced during the Task 3 implementation was effectively utilising both AVL trees and red-black trees simultaneously to organise and retrieve service request information. Both of these tree structures offer similar balancing capabilities, and the need to utilise both meant careful management of duplicated entries was required. More so, I needed to ensure that the user interface displayed the data efficiently without redundancy.

Another challenge I faced was the management of graphs and their traversal algorithms to optimise the relationships between service requests. The creation of meaningful service dependencies and ensuring they were displayed in the correct order was essential. To do so, I implemented Prim’s Minimum Spanning Tree Algorithm and Bread-First Search to effectively manage these dependencies.

Finally, the integration of these advanced data structures in the C# Windows Form environment required solving compatibility issues with UI controls such as the ListView component. Synchronising the advanced data structures with the user interface was a technical hurdle that I eventually overcame.

#### How These Challenges Were Overcome

To address the duplication challenge faced when working with both AVL trees and red-black trees, the design decision was made to ensure that only one unified view was displayed to the user whilst both trees stored and managed service requests. The backend would rely on both trees for different use cases. AVL trees for balanced searching by request ID and red-black trees for service request order consistency. But in the end, only one set of results would be shown to the user.

To effectively handle service request dependencies, careful management of the graph structure was utilised making use of adjacency lists and node management. The use of Prim’s Algorithm optimised dependencies efficiently. Providing minimal overhead when calculating which tasks needed to be completed first.

Lastly, to ensure the accurate functioning of the service request display system and allow for responsive and scalable handling of service requests, thorough testing and debugging of both the tree structures and graph traversal algorithms was performed.

#### Key Learnings

The completion of this project provided valuable lessons through the practical implementation of advanced data structures in real-world applications. The following are my key takeaways:

* Data structure selection: AVL trees and red-black trees both offer efficient balancing and fast lookups. However, through research into their subtle differences, I could effectively implement both structures to optimise different parts of the application.
* Graph theory: Concepts of relationship management in complex systems where service requests depend on each other, were reinforced through the implementation of graph traversal and minimum spanning tree algorithms.
* C# Windows forms integration: The skill of managing background data structures alongside frontend UI components like the ListView was important and necessary to ensure data displayed seamlessly while maintaining performance.
* Problem-solving skills: Extensive research was required to gather a detailed understanding of recursive functions, balancing techniques, and optimal traversal paths to perform debugging of tree and graph-related algorithms. These specific problem-solving skills were developed throughout the project.

Programming Techniques:

* Event-driven programming: Using event handlers in C# Windows forms was imperative to effectively respond to user interactions such as submitting requests, searching for events, and viewing statuses.
* Recursive functions: Recursion for insertion, deletion, and traversal was implemented for both AVL and red-black trees. Offering practical experience with recursion in balanced tree structures.
* Algorithm design: The implementation of Prim’s Minimum Spanning Tree Algorithm and Bread-First Search algorithms required careful design to ensure efficient traversal and minimal resource usage.

## Technology Recommendations

#### ASP.NET MVC Framework

The current municipal services application runs as a Windows Form application. By providing a web-based solution using ASP.NET MVC, accessibility would be improved for users, offering the ability to report issues, check events, and check service request statuses through any web browser. Resulting in increased citizen engagement as there will be no need to download and install software (ChatGPT, 2024).

Benefits (ChatGPT, 2024):

* Enhanced accessibility for users across different devices.
* Easier to scale and maintain as compared to a desktop-based application.
* Improved user experience with a modern web interface.

Compatibility: ASP.NET MVC can be seamlessly integrated with C# code. Existing logic for handling service requests can be reused with slight modifications to fit the web-based platform.

Azure Cloud for Data Storage and Processing

The adoption of a cloud platform such as Microsoft Azure would further enhance the scalability of the system. By potentially offloading the processing and storage demands from local machines. This would provide the large-scale deployment of the application. This can be particularly helpful in environments where multiple municipalities are using the system simultaneously (ChatGPT, 2024).

Benefits (ChatGPT, 2024):

* Scalable infrastructure with the ability to handle a large amount of service requests.
* Automatic backups, high availability, and disaster recovery features.
* Potential integration with AI services for predictive analytics on service request patterns.

Compatibility: Microsoft Azure provides seamless integration with .NET applications. Existing code and data structures can be ported to the cloud with little refactoring.

AI-Based Chatbot Integration

The addition of an AI chatbot to the municipal services application would provide users with real-time assistance with any uncertainties they may have while using the application. In turn, offering a more user-friendly application (ChatGPT, 2024).

Benefits (ChatGPT, 2024):

* Provides instant help to users, reducing the need for customer support.
* Uses natural language processing to improve user interaction.
* Enhances user engagement and satisfaction.

Compatibility: Frameworks like Microsoft Bot Framework offer AI chatbots that can be easily integrated with .NET applications. Making use of existing service request data to provide personalised responses.

To conclude, the implementation of said technologies can drastically improve the functionality, user experience, and scalability of the Municipal Services Application. Whilst keeping compatibility with the existing .NET framework codebase.

## ChatGPT Input/Output

**Input**: “For a Municipal Services Application that is a locally run desktop program used for reporting and tracking service requests, what additional technologies or tools could enhance its performance and user experience?”

**Output**: ”For a locally run Municipal Services Application focused on reporting and tracking service requests, the following technologies could improve performance, scalability, and user experience:

1. **ASP.NET MVC Framework**  
   Transitioning the application to an ASP.NET MVC web-based platform would allow users to access it via any browser, increasing accessibility beyond a single desktop environment. This shift would enable users to report issues, view updates, and receive notifications without requiring the application installation on each machine, making it more versatile and engaging.
2. **Azure Cloud for Data Storage and Processing**  
   Moving data storage and processing to Microsoft Azure would enhance the application’s scalability, allowing it to handle larger volumes of service requests and user data. Azure’s robust infrastructure supports automatic backups, high availability, and disaster recovery. It also offers built-in analytics and AI services, making it easy to implement data-driven insights into service patterns and municipal needs.
3. **AI-Based Chatbot Integration**  
   Adding an AI-powered chatbot, such as one using Azure Bot Services, would provide users with real-time support directly within the application. The chatbot could guide users through reporting issues, tracking request status, and finding event information. This integration would streamline the user experience and reduce response times by providing instant assistance and personalized responses based on user needs and history.

These technologies would modernize the Municipal Services Application, making it more scalable, accessible, and responsive to user needs while enhancing data management and overall efficiency.”

# References

ChatGPT. (2024). ChatGPT. San Francisco: OpenAI. Retrieved August 05, 2024, from https://chat.openai.com/