Author

Name: Joel
Surname: Shaduka
Student No: ST10041239
Group No: 1

Project Completion Report

Introduction

This report details the completion of the Municipal Services Application project, outlining the process of implementation, challenges encountered, and the solutions I developed throughout Parts 1, 2, and 3. Additionally, the report reflects on the insights gained during the project, particularly regarding the integration of advanced data structures to enhance functionality and user experience.

Project Overview

The general experience of developing the Municipal Services Application (MSA) can be broken down in to 3 phases with specific objectives and challenges that developed and aided each other to the final part.

1. **Part 1:** Focused on implementing the "Report Issues" feature, and mainly establishing the foundational architecture of the application, with an extra CSV writer & reader feature.
2. **Part 2:** Introduced advanced data structures to enhance the "Local Events and Announcements" feature while familiarising with DataGrids & advanced data rendering practices.
3. **Part 3:** Finalized the implementation with the "Service Request Status" feature, integrating advanced data structures such as Binary Search Trees (BSTs), Heaps, and Binary Trees.

Each phase required a progressive understanding of user interactions, architectural decisions, and data structure applications to create a scalable and user-friendly system.
This has benefited me by giving me confidence with using and *knowing* when/where to use these data structures - a core part of being a skilled Software Engineer.

Challenges and Solutions

1. Identifying the Role of Data Structures in Features

Challenge:
At the start of each phase, it was initially challenging to determine where specific data structures would fit within the features and how they would contribute to enhancing the user experience.

Solution:
I addressed this challenge by first mapping out the user flow roughly on paper for each feature. By visualizing the sequence of user interactions, I identified the points where data structures like BSTs, Binary Trees, or Heaps can be used to optimize functionality. I started each feature by implementing the UI, connecting it to the underlying logic, and then integrating the relevant data structure. This systematic approach allowed me to resolve ambiguities and effectively "work my way down" the problem from user input to the backend.

2. Structuring the Codebase Without MVC Enforced by Default

Challenge:
Windows Forms does not enforce the Model-View-Controller (MVC) architecture by default, like WinForms or ASP.NET, making it challenging to maintain a clean separation of concerns in the codebase. Additionally I've been using the MVC architecture in my other university & hobby projects using JavaScript, and it's really helped solve simple common issues and keeping my focus on the business logic and features. Not trying to loosely wire features things together

Solution:
To overcome this, I imposed an MVC-like structure by organizing the codebase into Services, Models, and Controllers (Page Code Behinds act as the Primary Controller), supported by consistent folder and file naming conventions. This approach facilitated the modular development of features, enabling easy registration of new pages and rapid integration of complex logic. For example, in Part 3, this setup allowed me to add a new page and update the UI in under ten minutes, demonstrating the scalability and flexibility of the architecture (*Model-view-controller architecture pattern: Usage, advantages, examples*).

3. Overcomplicating Solutions

Challenge:
As someone who tends to explore all possible solutions, I occasionally overcomplicated implementations by considering features beyond the project’s requirements. This led to inefficiencies in both planning and development.

Solution:
I addressed this by prioritizing simple, focused solutions that aligned with the project scope. By narrowing my focus, I improved both the quality and speed of my implementations. This shift in mindset not only enhanced productivity but also ensured that the solutions were practical and robust.

Insights from Each Phase

Part 1: Foundation and Architecture

The initial phase required critical decisions regarding the application's architecture. I opted for an MVC-inspired structure, which laid the foundation for a scalable and maintainable application. Developing the "Report Issues" feature also introduced me to designing user engagement strategies within a Windows Forms environment, where the balance between simplicity and functionality was key. It was during this implementation that I made the global `StateManager.cs` which allowed me to turn adding new pages into a simple, creating of files, and registering it to the StateManager.
This was challenging but rewarding, showing me how to manipulate architecture to tool being used.

Part 2: Applying Data Structures to User Features

This phase focused on enhancing the "Local Events and Announcements" feature by leveraging data structures such as Sorted Dictionaries and Queues. These structures enabled efficient categorization and retrieval of event data, significantly improving user interactions.

During this phase I learnt when & how to use event handlers effectively when implementing a `onHover` tracker for the DataGrid rows. Initially the idea seemed ambitious, but after a quick reference to the Windows Forms documentation I was able to implement the feature and surprise myself.
I also developed my skills in working with Windows Forms components such as DataGrids and ComboBoxes, learning how to dynamically populate these elements based on the underlying data.

Part 3: Advanced Data Structure Integration

The final phase was the most technically demanding, requiring the integration of advanced data structures like BSTs, Binary Trees, and Heaps for the "Service Request Status" feature. This phase deepened my understanding of real-world applications of these structures, particularly in optimizing sorting, searching, and filtering operations. For example, implementing a Min-Heap

for prioritizing service requests based on status demonstrated how theoretical concepts directly translate into practical solutions.
After completing Part 3, I know understand how all the various concepts learnt throughout my university degree connect together, irrespective that different languages were used. The core fundamentals are always present.

Key Learnings and Reflections

The project was a valuable learning experience that reinforced several core principles of software engineering:

1. **Systematic Problem-Solving:** By breaking down problems into manageable components, I was able to design and implement features systematically, avoiding unnecessary complexity.
2. **Architecture Matters:** The early decision to adopt an MVC-inspired structure paid dividends throughout the project, making the application easier to extend and maintain.
3. **Practical Application of Data Structures:** The project highlighted how different data structures—like BSTs, Binary Trees, and Heaps—solve specific problems efficiently in software systems.
4. **Focus on User-Centric Design:** Ensuring that features were intuitive and aligned with user needs remained a priority throughout development.

Conclusion

The Municipal Services Application project exemplified the challenges and rewards of designing a robust software solution. By overcoming initial hurdles in feature planning, architecture, and implementation, I developed a functional application that met the project requirements. The integration of advanced data structures not only enhanced performance but also provided a deeper understanding of their practical applications. This project has been a significant step forward in my journey as a software engineer, equipping me with the skills and insights necessary for future professional challenges.

Technology Recommendations

1. Transition to ASP.NET (Web)

Justification:

ASP.NET (Web) is a modern framework designed for building scalable, secure, and user-friendly web applications. Out of the box it provides **high performance, build-in security with its authentication features, interoperability, reliability and ease of maintaince.**
Transitioning from Windows Forms to ASP.NET offers the following advantages:

- **Enhanced User Accessibility:** A web-based application removes platform dependency, allowing users to access the application via any device with a browser, including smartphones, tablets, and desktops. This aligns with the goal of creating an inclusive municipal platform for all citizens.
- **Improved Developer Experience:** ASP.NET provides built-in support for modern web technologies, enabling rapid development and easier debugging. Its modular architecture simplifies feature integration, ensuring the scalability of the application.
- **Streamlined Integration with External APIs and Databases:** ASP.NET natively supports RESTful APIs and database integrations, making it easier to expand functionality, such as adding geolocation or notification features.
- **Cost-Effective Hosting:** By deploying the application on a cloud platform (e.g., Azure), the municipality can minimize infrastructure costs while maintaining high availability. (*ASP.NET: Key Components, benefits and when to use it* | *indeed.com*)

Benefits:

Switching to ASP.NET will significantly enhance the application's usability, allowing citizens to interact with the service on-the-go. For developers, it simplifies the process of maintaining and scaling the application, ensuring its sustainability as user demand grows. For stakeholders, it provides far more possibilities and opportunity of features & functionality the application can achieve.

2. Windows Presentation Foundation (WPF)

Justification:

If the project remains desktop-based, Windows Presentation Foundation (WPF) is a modern alternative to Windows Forms that offers a more advanced UI and improved development experience. Although WPF doesn't offer as many features & options as ASP.NET Web, it will still be a benefitical framework upgrade. The key features it will provide are:

- **Sophisticated User Interfaces:** WPF supports XAML for UI design, enabling dynamic, visually appealing, and interactive interfaces that enhance user satisfaction.
- **Simplified Data Binding:** WPF’s robust data-binding capabilities allow seamless integration between UI components and the underlying data, streamlining the implementation of features like filtering and sorting service requests.

- **Support for the MVVM Pattern:** WPF encourages using the Model-View-ViewModel (MVVM) architecture, which enhances code maintainability by ensuring clear separation of concerns.
  (*WPF vs winforms - which one is right for your project?* 2023)

Benefits:

WPF improves the application's aesthetics and functionality, by delivering a richer user experience. It also simplifies complex interactions, such as sorting and filtering requests, by leveraging built-in features like data templates and commands.

---

3. Relational Database (e.g., SQL Server)

Justification:

A relational database like SQL Server is essential for managing the growing volume of user data, such as service requests and categories. It provides:

- **Persistent Storage:** A database ensures data consistency and availability, even if the application shuts down or restarts.
- **Optimized Queries:** Complex operations like filtering, sorting, and searching can be efficiently handled by SQL queries, reducing the load on the application logic.
- **Data Relationships:** A relational model captures the relationships between data entities (e.g., requests, users, and statuses), supporting better data organization and retrieval.

Benefits:

Integrating SQL Server will enable the application to manage data efficiently, ensuring reliable performance as the user base expands. For example, storing and querying service requests by category or status will become faster and more scalable. Additionally database like PostgreSQL provide database level functions and re-usable queries that operate at the database levels, this will remove "unnecessary" additional business logic as data query complexities are managed at the database level.
The only side effect is that the database server must be available to users centrally, which may be undesirable to stakeholders.

4. Adopt Cloud Storage for Media Attachments (e.g., Azure S3 Storage)

Justification:

Currently, attachments (e.g., images or documents) may be stored locally or inefficiently. Cloud storage provides a scalable and secure solution. Services like Amazon S3 provide Role Based Authentication to uploaded files, fast delivery speeds as AWS has servers in Cape Town, and proven solutions for file management:

- **Global Accessibility:** Users can upload and retrieve files from any device without being tied to local storage.
- **Scalability:** Services like Azure S3 Storage can handle vast amounts of data, ensuring the application remains responsive under heavy load.
- **Data Security:** Built-in encryption and access controls ensure that user data is protected from unauthorized access.
- **Low-Cost**: Services like AWS S3 offer their cloud solutions at affordable prices, especially for those experiencing heavy data loads, this will be desirable to the Municipality in keeping operational costs low for the application.
  (*What is AWS S3: Overview, features & Storage Classes explained | Simplilearn*)

Benefits:

Using cloud storage simplifies file management and enhances the application's performance. For example, users submitting images with their service requests can upload files seamlessly, while the system ensures data security and availability.

5. Implement a Message Queue System (Azure Queues)

Justification:

A message queue system can improve the responsiveness and reliability of features like notifications or status updates. These message queues, or even cloud functions, allows the application infrastructure to remain light weight and secure, as the core business logic is contained within the message queues (saving data to a database or sending automated notifications):

- **Asynchronous Processing:** Notifications can be sent asynchronously, ensuring the UI remains responsive even during high system activity.
- **Reliable Delivery:** Messages are queued and processed in order, guaranteeing that updates are delivered even if the system experiences downtime.
- **Scalability:** A message queue system can handle a large volume of messages without affecting application performance.
- **Low-Cost**: Most Cloud providers charge for this feature by the amount of execution time of the message queue, which allows the developer to focus on writing efficient code &

using appropriate data structures to ensure costs are low.
(Guide, 2024)

Benefits:

Implementing a message queue will enhance the user experience by delivering real-time updates on service requests. For instance, users can receive instant notifications about changes in request statuses without delays. Additionally, these queues can be used exclusively for delivering real-time functionality like notifications and messaging, allowing the applications features to grow.

6. Integrate External APIs

Justification:

External APIs can extend the application's functionality and provide new, value-added features:

- **Geolocation Services:** APIs like Google Maps can help users provide precise locations for reported issues, improving the accuracy of the data.
- **Weather Data:** Weather APIs can add context to local announcements, such as event planning or service schedules affected by weather conditions.
- **Communication Tools:** APIs like Twilio or SendGrid can enable SMS or email notifications, ensuring users are informed about service updates.

Benefits for the Application:

External APIs improve the application's user experience by adding contextually relevant features. For example, integrating Google Maps allows users to pinpoint their location when reporting issues, reducing errors and enhancing efficiency. These features paired with the technologies above allows for a robust and feature-full application that will please both the stakeholders, and citizens (users)

Summary of Recommendations

Technology	Key Benefit
ASP.NET (Web)	Improves accessibility, scalability, and developer experience
Windows Presentation Foundation (WPF)	Enhances UI design and integrates better with modern patterns (MVVM)

Technology	Key Benefit
Relational Database (SQL Server)	Ensures persistent and efficient data management
Cloud Storage	Simplifies media file handling with secure and scalable solutions
Message Queue (RabbitMQ)	Enables asynchronous processing and reliable notifications
External APIs	Adds features like geolocation, weather data, and notifications

Lecture Feedback

Please not that I did not recieve any feedback for my submitions. Therefore no further enhancements for Part 1 & 2 have not been implemented directly. However I did do some general house keeping in the codebase.

Bibliography

(No date a) *What is AWS S3: Overview, features & Storage Classes explained | Simplilearn*. Available at: https://www.simplilearn.com/tutorials/aws-tutorial/aws-s3 (Accessed: 18 November 2024).

(No date b) *What is AWS S3: Overview, features & Storage Classes explained | Simplilearn*. Available at: https://www.simplilearn.com/tutorials/aws-tutorial/aws-s3 (Accessed: 18 November 2024).

(No date c) *ASP.NET: Key Components, benefits and when to use it | indeed.com*. Available at: https://www.indeed.com/career-advice/career-development/what-is-aspnet (Accessed: 18 November 2024).

Guide, T.T. (2024) *Azure Queues, a fundamental component of Microsoft Azure's cloud services, play a pivotal role in...*, *Medium*. Available at: https://medium.com/@tttechiesguide/azure-queues-a-fundamental-component-of-microsoft-azures-cloud-services-play-a-pivotal-role-in-eaa987eb858d (Accessed: 18 November 2024).

Model-view-controller architecture pattern: Usage, advantages, examples (no date) *HackerNoon*. Available at: https://hackernoon.com/model-view-controller-architecture-pattern-usage-advantages-examples (Accessed: 18 November 2024).

WPF vs winforms - which one is right for your project? (2023) *ByteHide*. Available at: https://www.bytehide.com/blog/wpf-vs-

winforms#:~:text=WPF%20provides%20a%20rich%20set,or%20powerful%20as%20WPF's%20implementation. (Accessed: 18 November 2024).