

ST10096757
Monique Jackson
PROG POE

IMPLEMENTATION DOCUMENT

INTRODUCTION:

The final submission for the PROG POE, asks the developer to develop the functions related to the "Service Request Status" for the Municipality desktop application. It is a requirement of this feature to incorporate the use of tree algorithms, heaps and graphs - to name a few - to create a more efficient program; these will be viewed in detail in this document.

GETTING STARTED:

** PREREQUISITES **

- ★ Install Git (If not already installed)

Link to download: <https://git-scm.com/downloads>

- ★ Install Visual Studio version 19 (If not already installed)

Link to download:

<https://visualstudio.microsoft.com/vs/older-downloads/>

** CLONING THE PROJECT **

1. Launch Visual Studio on the chosen device.
2. In Visual Studio, go to **`File > Start Window`**, if it does not automatically open there.
3. On the Start Window select **`Clone a repository`**.
4. Next, go to the GitHub repository that must be cloned.
5. Copy that repos URL.

In this case, the URL is:

https://github.com/ST10096757/ST10096757_MoniqueJackson_MunicipalityApp_POE

6. Paste the URL in the **Repository location** field in Visual Studio.
7. After pasting the URL, choose a file path to store the project.
Either click the Browse button and choose a directory, or keep it as the default.
8. After the path has been set, click the **Clone** button.
9. When the cloning process is complete, the project will automatically open in Visual Studio.

RUNNING THE APPLICATION

** PREREQUISITES **

- ★ WPF .NET Framework 4.8 or later
- ★ Newtonsoft.Json Nuget Package - the latest version
- ★ service_requests.json file

Should the nuget package not be installed, navigate to the "Manage NuGet Packages" and search for the package name and select install. The `service_requests.json` should already be stored within the package in the 'Resources' folder.

** RUNNING THE APPLICATION **

1. On the top task bar, go to the '**Build**' menu tab, and select **Build Solution**.
2. Select the '**Start**' button

FEATURES:

★ PART 1 (REPORT ISSUES)

Location Input: Specify the location of the report issue.

Category Selection: Dropdown to select the issue category

Description Box: Provide detailed description using a RichTextBox

Media Attachment: Attach image or documents related to issue

Progress Tracking ("User Engagement Feature"): A progress bar that updates based on form completeness.

★ PART 2 (LOCAL EVENTS & ANNOUNCEMENTS)

Event Display: View upcoming events with details

Search Functionality: Filter events by category or date

Recent Searches: View past search queries

Recommendations: Get event recommendation based on user preference

Clear Filters: Resets all filter and search settings

★ POE (SERVICE REQUEST STATUS)

Service Request Graphs: A graph that shows the service requests and their dependencies.

Minimum Spanning Tree: Computes and displays the relationship between service requests

Service Request Filtering: Filter service request through priority (Heap) or status

Search by Request ID: Displays data for request ID using Binary Search Tree

Sorting by Submission Date: Sorts requests by submission date using Red Black Tree

MENU FEATURES:

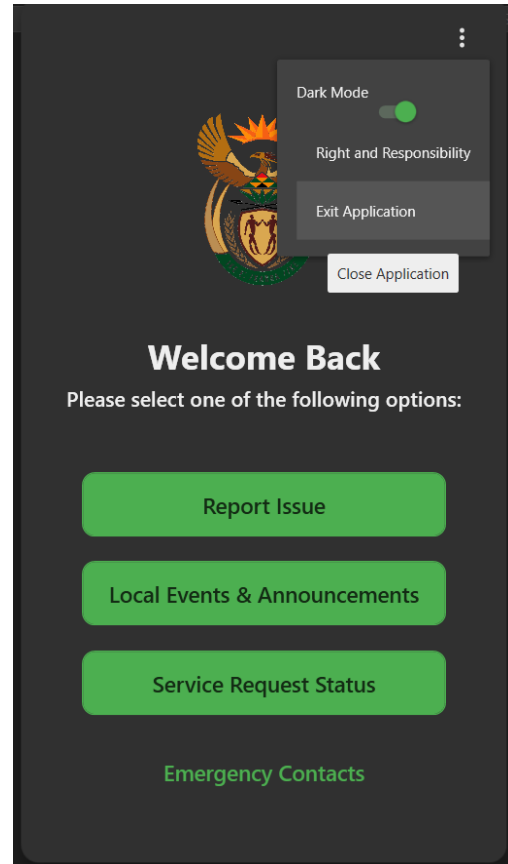
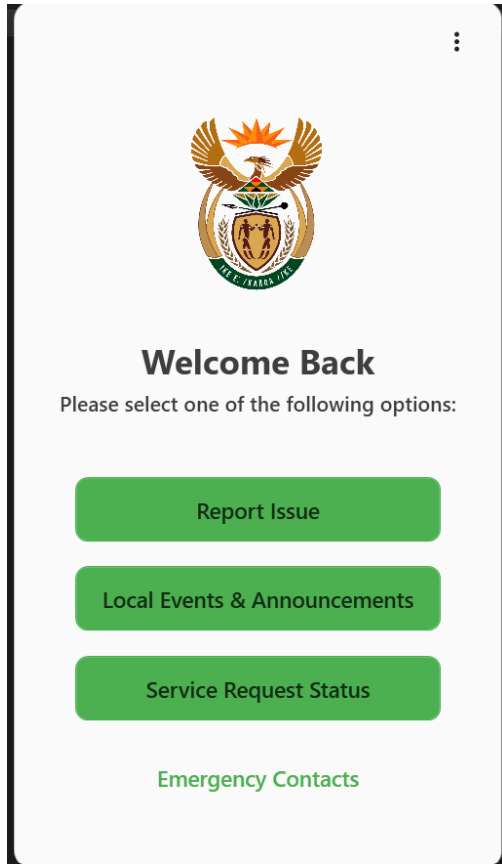
Theme Toggle: Switch between light and dark themes.

Emergency Contacts: Access emergency contact information easily.

Bill of Responsibilities: Read about user responsibilities as a South African.

NAVIGATION:

The apps opens on the Main Menu:



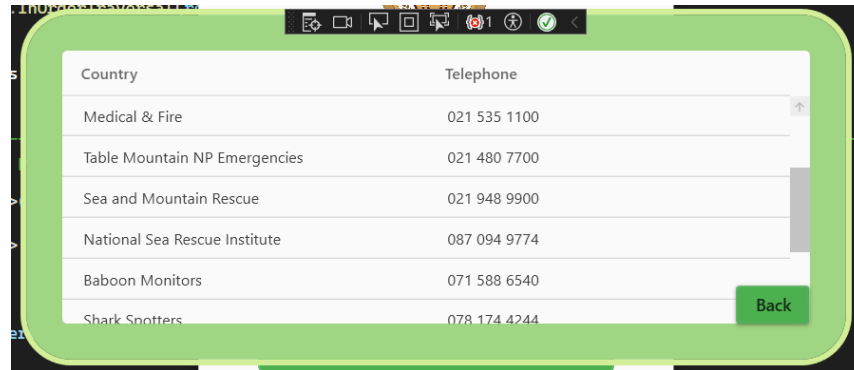
The main menu houses the 3 functions of the app:

- Report Issues (Part 1)
- Local Events & Announcements (Part 2)
- Service Request Status (POE)

On the main, if the user selects the `vertical 3 dots` on the right side of the screen, they will be presented with the option to:

- Toggle between light and dark mode
- Open a document where they can read up upon their responsibility as South Africans
- Exit the application

At the bottom of the main menu, there's an `Emergency Contacts` button. This navigates the user to an interface that holds South African emergency contacts.



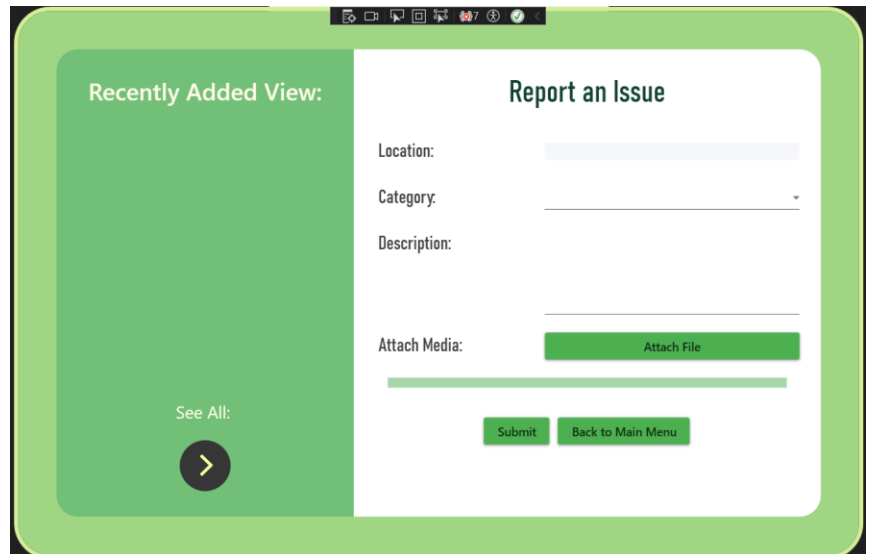
Country	Telephone
Medical & Fire	021 535 1100
Table Mountain NP Emergencies	021 480 7700
Sea and Mountain Rescue	021 948 9900
National Sea Rescue Institute	087 094 9774
Baboon Monitors	071 588 6540
Shark Spotters	078 174 4244

★ PART 1 - REPORT ISSUES:

This is the Report Issues Interface:

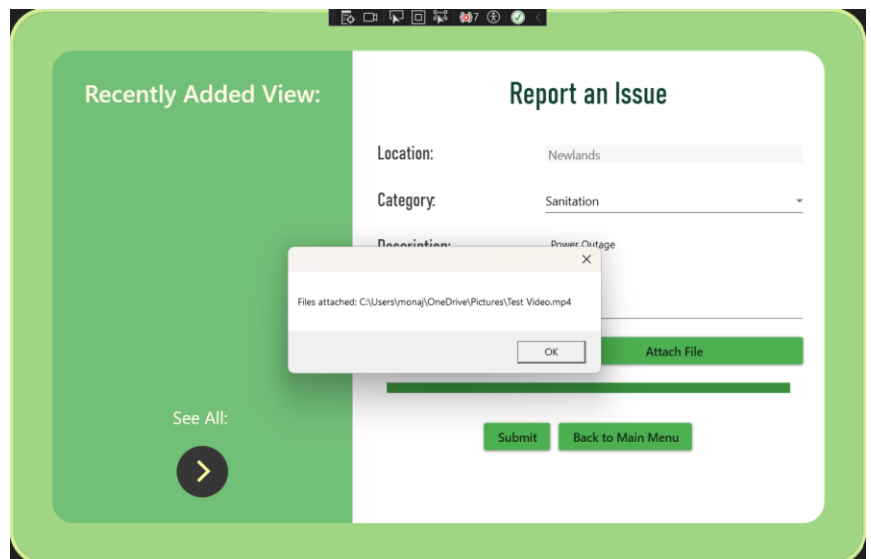
Users need to fill in the required information which is all the fields.

The “user engagement feature” progress bar will update as each field is filled.

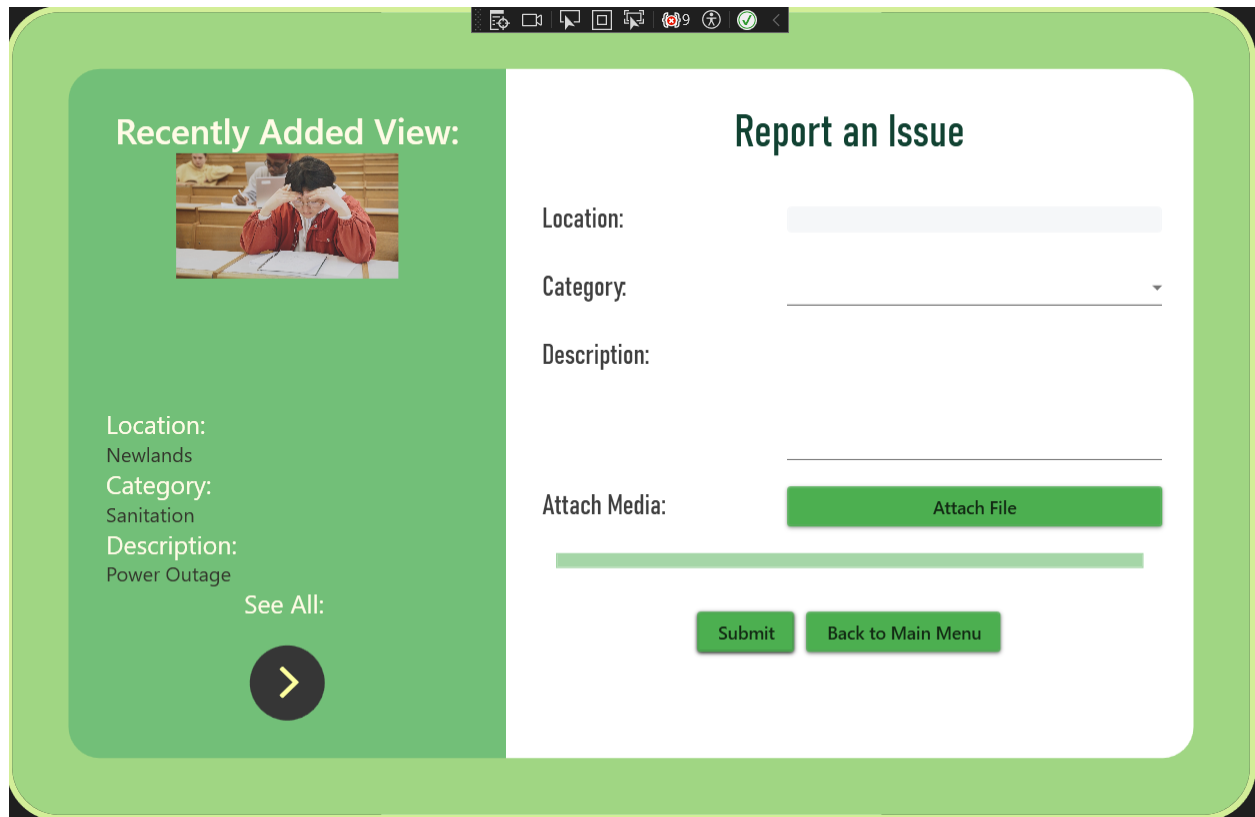


To attach media, the user must select the `Attach File` button

When a user uploads media to accompany their issue, this message appears



The user must then select the 'Submit' button and what they've just entered will display on the left hand side of the interface.

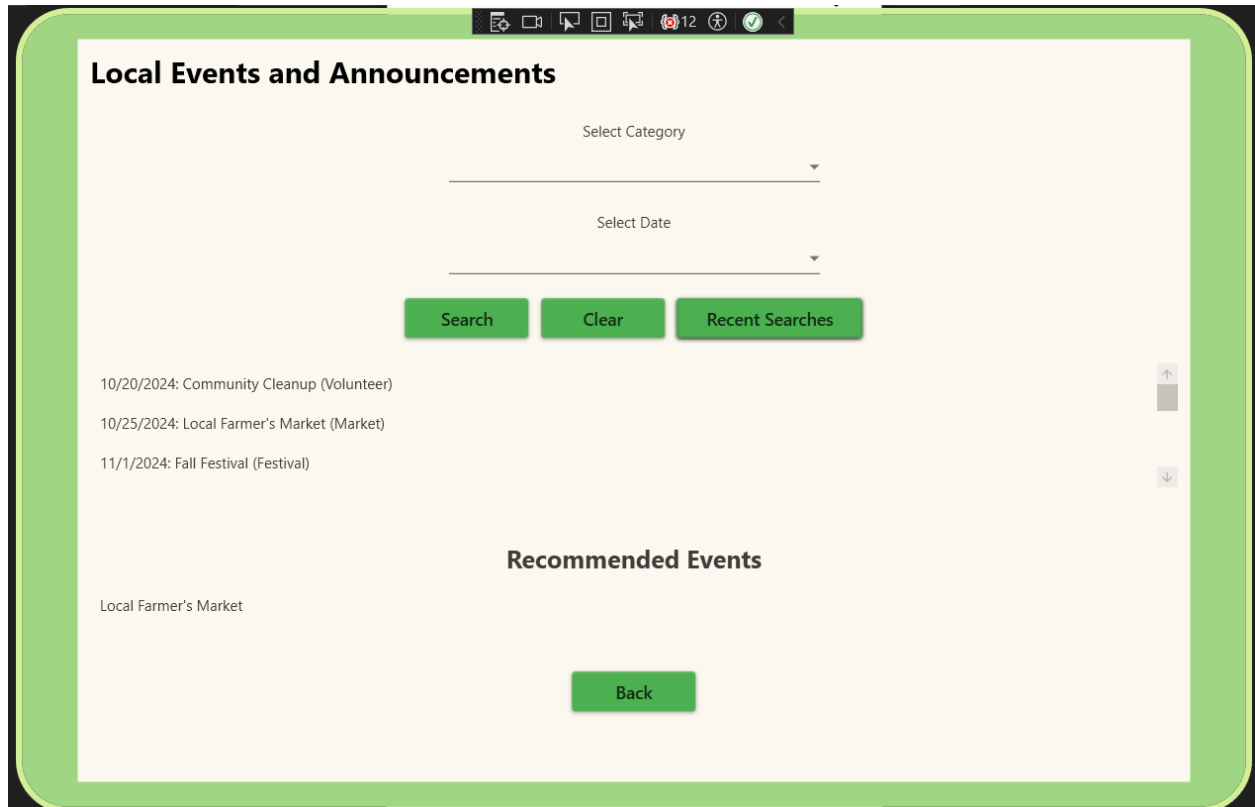


The screenshot shows a web application interface with a green header and a white main content area. On the left, there is a green sidebar titled 'Recently Added View:' which contains a photo of a person in a red jacket sitting at a desk, and text details: 'Location: Newlands', 'Category: Sanitation', and 'Description: Power Outage'. Below this text is a 'See All:' link with a yellow arrow icon. The main content area is titled 'Report an Issue' and contains a form with the following fields: 'Location:' (text input), 'Category:' (dropdown menu), 'Description:' (text area), and 'Attach Media:' (file upload button labeled 'Attach File'). At the bottom of the form are two buttons: 'Submit' and 'Back to Main Menu'.

Should the user wish to return to the main menu, they must simply select the 'Back to Main Menu' button.

★ PART 2: LOCAL EVENTS & ANNOUNCEMENTS

When the user selects the 'Local Events & Announcements' button on the main menu, they are navigated here:



Users can filter through the list of events by either selecting a category or a date from the drop down boxes, and clicking the `Search` button.

To clear the selected category or date, the user should click the `Clear` button.

When the user selects the `Recent Searches` button and it will display an event that matches their recent search and recommends it to the user.

To return to the main menu, the user must select the `Back` button.

★ POE - SERVICE REQUEST STATUS

When the user selects the `Service Request Status` button on the main menu, they are navigated here:

	Resident Name	Contact Detail:	Request T	Description	Status	Request
1	Monique Jackson	monique.j@example.cor	Streetlight Repair	The streetlight in front of my hou	Pending	2023-11-16
2	Jake Davis	jake.d@example.com	Pothole Repair	There is a large pothole on Oak S	In Progress	2023-11-15
3	Anna Smith	anna.s@example.com	Water Leak	There is a water leak in the basen	Completed	2023-11-14
4	Lindiwe Nkosi	lindiwe.nkosi@example.	Streetlight Repair	Streetlight outside my shop is flic	Pending	2023-11-17
5	Sipho Mthembu	sipho.mthembu@exam	Sewer Blockage	The sewer line near my house is l	In Progress	2023-11-18
6	Thabo Malema	thabo.malema@exampl	Pothole Repair	Multiple potholes along Main Rd	Pending	2023-11-19
7	Rachel Mabuza	rachel.mabuza@exampl	Streetlight Repair	The streetlight on the corner of n	In Progress	2023-11-20
8	Zanele Dlamini	zanele.dlamini@exampl	Water Leak	A small water leak in the backyar	Completed	2023-11-21
9	Kabelo Mokoena	kabelo.mokoena@exam	Sewer Blockage	Overflowing sewer in my area.	In Progress	2023-11-17
10	Mpho Motsepe	mpho.motsepe@examp	Pothole Repair	The potholes on the highway nee	Pending	2023-11-19

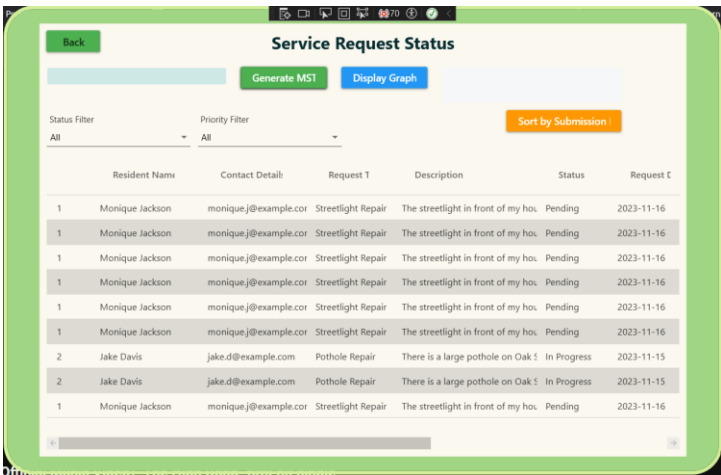
When a user enters a numeric value in the textbox, it will search for the request by that number, should that number be that same as a Request ID saved in the JSON file.

	Resident Name	Contact Detail:	Request T	Description	Status	Request I
4	Lindiwe Nkosi	lindiwe.nkosi@example.	Streetlight Repair	Streetlight outside my shop is flic	Pending	2023-11-17

The user can filter through the data by selecting a status in the status drop down box.



Users can do the same with priority, and be presented data based on priority level.



When the user selects the 'Generate MST', the data displayed shows the relationship between the requests.

Display Graph

Edge: 1 -> 3 with weight 0

Edge: 1 -> 5 with weight 0

Edge: 1 -> 6 with weight 0

When the user selects the 'Display Graph' button, in the little display, it

shows the requests and their dependencies.

Lastly, the user can filter the data by selecting the orange `Sort by Submission` button, which will display data from first submitted to last submitted.

CODE STRUCTURE:

★ PART 1 - REPORT ISSUES

ReportIssues_UC.xaml: Defines the user interface for the issue reporting control.

ReportIssues_UC.xaml.cs:

Contains the logic for handling user interactions, including form submission, media attachment, and progress tracking.

MainWindow.xaml: The main window that hosts the ReportIssues_UC user control.

MainWindow.xaml.cs: Manages navigation and other main window functionalities.

ViewIssue_UC.xaml: Defines the user interface for viewing reported issues.

ViewIssue_UC.xaml.cs: Contains logic for displaying and interacting with the list of reported issues.

IssuesClass: The data model for reported issues.

IssuesViewModel: The view model for the application, managing the reported issues and allowing data binding between the UI and the data model.

★ PART 2 - LOCAL EVENTS & ANNOUNCEMENTS

MainWindow.xaml: The main window that has user controls and manages navigation.

MainWindow.xaml.cs: Contains the logic for theme management, event handling, and user control navigation.

Events_Announce_UC.xaml: Defines the user interface for the events announcement control.

Request ID	Description	Status	Request Date	Priority	Assigned To	Location
Water Leak	There is a water leak in the baser	Completed	2023-11-14	High	Tom Green	789 Pine St., Dur
Pothole Repair	There is a large pothole on Oak S	In Progress	2023-11-15	Medium	Sarah Lee	456 Oak St., Pre
Pothole Repair	There is a large pothole on Oak S	In Progress	2023-11-15	Medium	Sarah Lee	456 Oak St., Pre
Streetlight Repair	The streetlight in front of my hou	Pending	2023-11-16	High	John Doe	123 Elm St., Cape
Streetlight Repair	The streetlight in front of my hou	Pending	2023-11-16	High	John Doe	123 Elm St., Cape
Streetlight Repair	Streetlight outside my shop is flc	Pending	2023-11-17	Medium	Musa Khumalo	11 Market St., Jol
Streetlight Repair	Streetlight outside my shop is flc	Pending	2023-11-17	Medium	Musa Khumalo	11 Market St., Jol
Sewer Blockage	Overflowing sewer in my area.	In Progress	2023-11-17	High	Lindiwe Mthembu	23 Church St., Kir
Sewer Blockage	Overflowing sewer in my area.	In Progress	2023-11-17	High	Lindiwe Mthembu	23 Church St., Kir
Sewer Blockage	The sewer line near my house is l	In Progress	2023-11-18	High	Nandi Zulu	34 Blouberg Rd.,

Events_Announce_UC.xaml.cs: Contains the logic for handling event display, search functionality, and recommendations.

EventsClass: Has the data model for events.

EventManager: Manages events and provides methods for adding and retrieving events.

EventSearchManager: Handles recent searches.

RecommendationManager: Gives event recommendations based on user interactions.

IssuesViewModel: Manages data and logic for reporting issues.

★ POE - SERVICE REQUEST STATUS

MainWindow.xaml: Main window hosting user controls for navigation.

MainWindow.xaml.cs: Handles the logic for navigation, theme management, and user control switching.

ServiceRequestViewModel.cs: The **ViewModel** for handling service request logic, including searching, filtering, and generating the MST.

ServiceRequestManager.cs: Loads service request data from JSON and manages service request creation and manipulation.

ServiceRequest.cs: Defines the **ServiceRequest** data model.

Graph.cs: Defines the **Graph** data structure used for storing service requests and their dependencies.

Edge.cs: Defines the edges of the graph, representing dependencies between service requests.

RedBlackTree.cs: A tree structure used to sort service requests by submission date.

BinarySearchTree.cs: A tree structure used for quick lookup and sorting by ID.

MaxHeap.cs: A heap used for prioritising service requests based on urgency.

DATA STRUCTURES:

★ Data Structure 1: Max Heap

Type: Heap (Max Heap)

Purpose: The Max Heap is used to store service requests in a way that prioritises requests based on urgency or priority. It helps to

efficiently retrieve the highest-priority request without scanning through the entire list.

Role in Efficiency: The Max Heap makes for efficient retrieval and insertion of the highest-priority service request (GeeksforGeeks, 2024). Operations such as insertion and removal have a time complexity of $O(\log n)$, making it efficient for real-time prioritisation. The space complexity is $O(n)$, where n is the number of service requests (GeeksforGeeks, 2024).

Example: In the context of the application, the heap is used to insert service requests based on their priority (e.g., High, Medium, Low). When a new service request is reported, it is inserted into the heap, and when the application needs to process the highest priority request, it can do so in $O(\log n)$ time by simply removing the root of the heap.

★ Data Structure 2: Graph

Type: Graph (Adjacency List)

Purpose: The Graph is used to represent service requests and their dependencies or relationships. Each node in the graph corresponds to a service request, and edges between nodes represent dependencies (e.g., related issues or tasks that need to be completed together).

Role in Efficiency: The Graph allows for efficient traversal of service request relationships and supports operations like finding dependencies between requests (GeeksforGeeks, 2024). Using an adjacency list representation, the time complexity for finding all dependencies of a service request is $O(n)$, where n is the number of related requests (GeeksforGeeks, 2024).

Example: When adding a service request to the graph, the program checks if any other requests are related to it (e.g., a road repair request that might depend on sanitation work being completed first). The graph structure allows efficient look-up and maintenance of these relationships.

★ Data Structure 3: Minimum Spanning Tree (MST)

Type: Tree (Minimum Spanning Tree)

Purpose: The Minimum Spanning Tree (MST) is used to calculate the most efficient way to connect all service requests, based on their dependencies and priorities. The MST is built by considering the edges between service requests, where the weight of an edge could represent the cost or importance of a connection.

Role in Efficiency: The MST helps reduce the number of dependencies that need to be processed, ensuring that only the necessary connections are maintained (GeeksforGeeks, 2023). The time complexity of generating an MST using algorithms like **Kruskal's** or **Prim's** is $O(E \log V)$, where **E** is the number of edges and **V** is the number of vertices (service requests). This is more efficient than processing all possible connections in a dense graph (GeeksforGeeks, 2023).

Example: The MST is used to display the most important or cost-effective relationships between service requests. For instance, if a series of repair requests are related by geographic proximity or dependency, the MST ensures that only the most critical connections are visualised, aiding in decision-making.

★ Data Structure 4: Binary Search Tree (BST)

Type: Tree (Binary Search Tree)

Purpose: The BST is used to store service requests in a way that allows for fast look-up and insertion by **Request ID**. It ensures that requests can be efficiently found or inserted without needing to scan the entire list of service requests.

Role in Efficiency: The BST supports $O(\log n)$ time complexity for searching, inserting, and deleting requests, as long as the tree remains balanced (GeeksforGeeks, 2024). This makes it highly efficient for operations that require frequent look-up by request ID. Its space

complexity is $O(n)$, where n is the number of service requests (GeeksforGeeks, 2024).

Example: When a user searches for a specific service request by ID, the BST ensures that the search operation is fast (in $O(\log n)$ time). This makes finding a request by ID significantly faster than a linear search, especially when the application handles a large number of requests.

★ Data Structure 5: Red-Black Tree (RBT)

Type: Tree (Red-Black Tree)

Purpose: The Red-Black Tree is used to sort service requests by **Submission Date**, ensuring that requests are ordered chronologically. This tree is self-balancing and allows efficient sorting and retrieval of service requests based on their submission date.

Role in Efficiency: Like the BST, the RBT supports $O(\log n)$ time complexity for insertion, deletion, and search operations. However, the RBT guarantees a more balanced structure, which prevents performance degradation in the worst-case scenarios. The space complexity is $O(n)$, where n is the number of service requests.

Example: When service requests are loaded into the application, they are inserted into the Red-Black Tree based on their submission date. This ensures that when a user requests a list of service requests sorted by date, the application can retrieve them in $O(\log n)$ time, even if there are thousands of requests.

Authors

Signing off for the last time, this had been Monique Jackson.

Thank you for "READ"ing ME.

References:

GeeksforGeeks. (2024). *Heap Data Structure*. Available at: <https://www.geeksforgeeks.org/heap-data-structure/> [Accessed 18 November 2024].

GeeksforGeeks. (2024). *Graph Representation | Set 1 (Adjacency List)*. Available at: <https://www.geeksforgeeks.org/graph-representation-using-adjacency-list/> [Accessed 18 November 2024].

GeeksforGeeks. (2023). *Kruskal's Algorithm for MST*. Available at: <https://www.geeksforgeeks.org/kruskals-algorithm-for-minimum-spanning-tree/> [Accessed 18 November 2024].

GeeksforGeeks. (2024). *Binary Search Tree*. Available at: <https://www.geeksforgeeks.org/binary-search-tree-data-structure/> [Accessed 18 November 2024].

GeeksforGeeks. (2024). *Red-Black Tree*. Available at: <https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/> [Accessed 18 November 2024].