Qhawe Ngcongo
ST10283124
PROG7312

Programming POE

Updates Made Based on Lecturer Feedback

1. Functional Improvements

Area	Feedback	Update Implemented
Recommendation Feature	The recommendation system was missing or not functioning.	Implemented a RecommendationEngine class that suggests the most urgent or frequently reported issues using priority weighting and graph traversal logic. It now interacts with the service requests and displays the top weighted insights dynamically.
Service Request Status Tracking	Service requests lacked proper updating and visualization.	Enhanced the ServiceStatusForm to include real-time status tracking, color-coded statuses (Submitted, InProgress, OnHold, Resolved, Rejected), and weighted ranking insights.
Simulated Requests	Repeated IDs appeared and statuses defaulted to “Submitted”.	Added automatic ID synchronization (_nextId = allRequests.Count + 1) to prevent duplicates and random or user-selected status assignment when adding simulated requests.
Integration Between Pages	Limited communication between Report Issue, Events, and Status pages.	Implemented shared data handling through a static repository (SharedReportRepository) allowing submitted reports to populate the Service Request Status page instantly.

2. Data Structure Enhancements

Area	Feedback	Update Implemented
Sets Not Working	The set data structure was missing or unused.	Implemented a CustomSet to store and filter unique categories, locations, and report priorities, ensuring no duplicate entries.
Dictionary Unused	The dictionary was implemented but not integrated.	Integrated the CustomDictionary to manage event and announcement records, providing fast category-based lookup and mapping for improved retrieval.
Queue/Stack Implementation	The lecturer raised issues with EventManager.queue and AnnouncementManager.queue.	Fixed both by properly enqueuing and dequeuing items, ensuring correct FIFO order for announcements. The StackLogHistory was refined to store command and message logs with undo/redo simulation.
Graph, Tree, and Heap Usage	Graph and tree data structures were present but underutilized.	Strengthened integration by using the GraphSR for BFS traversal and MST calculations in the insights panel, AVL/Red-Black Trees for fast search, and MinHeapSR for urgency ranking.

3. User Interface (UI/UX) Enhancements

Area	Feedback	Update Implemented
UI Needed Polish	The interface looked basic and crowded.	Redesigned all forms with a consistent blue-and-white municipal theme, improved spacing, and standardized fonts (Segoe UI 10F). Alternate row shading and priority color coding improve readability.
Beginner-Friendly Interaction	UI did not support less tech-savvy users.	Added placeholder text, input validation, and tooltips to guide users. Also introduced auto-focus and placeholder reset logic to improve accessibility.
Filtering and Search	Filtering should be added.	Implemented search by ID, category, or location and basic filters for viewing specific statuses. Filtering logic dynamically refreshes the ListView without restarting the form.
Dynamic Insights Panel	Data presentation was minimal.	Created an Insights (Heap + Graph) section displaying BFS traversal, MST cost, and weighted request ranking in real-time using a TextBox formatted with monospace font for clarity.

4. Performance & Reliability Updates

Area	Feedback	Update Implemented
Crashes & Errors	Occasional runtime issues and unsupported property errors.	Replaced PlaceholderText (unsupported in older .NET) with manual placeholder handling, preventing compatibility errors.
Scalability	Limited modularity made updates difficult.	Reorganized project into clear folders (Models, Data, DataStructures, Logic, App_Config) improving maintainability and readability.
Data Persistence Simulation	Lack of persistent storage.	Added future-ready comments and structure for SQLite or Azure integration, simulating database logic through in-memory structures.

5. Documentation & Testing Improvements

Area	Feedback	Update Implemented
Limited Explanation of Data Structures	Descriptions were missing or unclear.	Added detailed XML documentation headers explaining the purpose and function of each data structure (ArrayListSR, GraphSR, AVLTreeSR, etc.).
Testing Coverage	Limited evidence of debugging or verification.	Conducted extensive manual testing on all buttons and functions (Add, Update, Refresh, Search, Simulate). Debugging ensured all forms load correctly and insights reflect updated data.
Code Readability	Code organization inconsistent.	Cleaned and formatted all methods with clear naming conventions, indentation, and grouping (UI setup, data handling, insights, etc.).