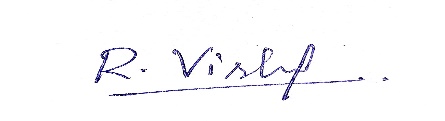
**Student Portfolio**

|  |  |
| --- | --- |
| **VISHAL R** | |
|  | **Register Number: RA2311056010028**  **Mail ID: vr5260@srmist.edu.in**  **Department: Data Science and Business System**  **Branch: CSE (Data Science)**  **Year / Sem/ Section: III / 3 /AL1** |
| **Subject Title: 21CSC201J Data Structures and Algorithms**  **Handled By: Dr. K. Rajkumar** | |
| **Email:** [**vishal25r07@gmail.com**](mailto:vishal25r07@gmail.com)  **LinkedIn:** <https://www.linkedin.com/in/vishal-r-690631336/>  **GitHub:** [**https://github.com/VISHAL-RAJA-25**](https://github.com/VISHAL-RAJA-25)  **Portfolio Website (if any):** https://sites.google.com/srmist.edu.in/vishalportfolio/home | |
| **ELab Completion Status** | |
| **Lab Experiment Completion status** | |
| **Real World Application in DSA: Chat application message History**  **Overview:**  In a chat application with message history, messages are exchanged between users in real time and are stored in chronological order. This allows users to access past conversations, ensuring continuity and easy reference to previous messages.  **Queue Representation**  **Elements:** Each element in the queue represents a message.  **Attributes:** Includes sender ID, message content, timestamp, and status (e.g., sent, delivered, read).  **Operations:**   * **Enqueue:** Adds a new message to the queue as it is sent by the user. * **Dequeue:** Removes a message from the queue after it has been acknowledged as read or archived. * **Peek:** Allows viewing the latest message in the queue without removing it, useful for notifications.   **Algorithm Implementation:**   * **FIFO (First In, First Out):** Ensures messages are processed in the order they were sent to maintain conversational flow. * **Priority Queue for Important Messages:** For cases where certain messages (e.g., urgent system alerts) need to be prioritized over others. * **Dynamic Queue Management:** Supports deletion or editing of messages, real-time status updates, and message archiving to keep the queue manageable and up-to-date.   **Key Components**  **Array/Linked List-Based Queues**   * **Array:** Fixed size, simple implementation, suitable for basic chat systems with limited message storage. * **Linked List:** Flexible and ideal for handling an unknown or variable number of messages, especially in active group chats.   **Circular Queue**  Reuses memory efficiently in static arrays by wrapping around, helpful in chat applications where older messages are overwritten after reaching storage limits.  **Priority Queue**  Prioritizes specific message types, like urgent notifications or alerts, ensuring they’re delivered and displayed promptly.  **Dynamic Updates**  Supports real-time changes such as message edits, deletions, and status updates (e.g., seen, delivered), keeping the chat history current.  **Skills Demonstrated**   * **Algorithm Design:** Implementing optimized queue operations for orderly and efficient message handling. * **Data Structure Analysis:** Choosing the appropriate queue type for effective memory and performance management in message storage.   **Programming Proficiency:** Advanced manipulation of queues for real-time message updates and prioritization.   * **User Interface:** Developing clear and intuitive chat interfaces for seamless user interaction with message histories.   **This approach highlights how queue-based systems provide scalable and reliable solutions for chat application message history.**  **GATE Questions Solutions:**  **https://srmist-my.sharepoint.com/:w:/g/personal/vr5260\_srmist\_edu\_in/EWKERcFBaqJBvYaQjbyk5nEBJnBDcgBmIdgDRUl2kst4hA?e=OgbL5p** | |
| **CERTIFICATIONS** | |
| **CODING COMPETETIONS**  **(Hacker Rank, Code Chef)** | |



**Signature of the Student**