**4-2 Milestone Three: Enhancement Two: Algorithms and Data Structures - Narrative**

Matthew R. Chandler

matthew.chandler2@snhu.edu

Southern New Hampshire University

CS 499: Computer Science Capstone

Instructor Joseph Conlan

October 1, 2025

**4-2 Milestone Three: Enhancement Two: Algorithms and Data Structures - Narrative**

The Android event tracking app that I used for category one is the same artifact I'm using for category two. It started out as my CS 360 project, which I improved upon in CS 499 with an emphasis on the data structures and algorithms that drive reminder scheduling. I created a scheduler for this milestone that arranges upcoming reminders in a priority structure and only keeps the next reminder loaded in the operating system. Through the use of a data structure that exhibits predictable behavior as the number of events increases, the timing logic was moved from scattered activity code to a single component.

This item was chosen for my ePortfolio because it shows that I can select and apply an appropriate data structure to a real feature while taking accuracy and performance into account. The primary improvement is a scheduler that uses a min-heap data structure to create a scheduler in place of the existing event alarms. The app now only sets the OS alarm for the earliest reminder stored in a heap keyed by trigger time. When it goes off, the app removes it from the heap, sends the SMS, determines the next reminder if one exists, puts it back into the heap, and sets the alarm for the newly determined reminder. This results in fewer wake-ups and reminders remaining on schedule.

I met the outcomes I planned to meet for this category. In Module One I stated that the algorithms and data structures enhancement would use a min heap to manage reminders and that goal is now satisfied in the running app. The update maintains the user visible behavior while improving the efficiency and predictability of the underlying logic. It aligns with the outcomes for designing and evaluating computing solutions with appropriate algorithmic principles and for communicating those choices in a clear way through the accompanying design narrative. I don't have any updates to my outcome-coverage plans.

Improving the artifact taught me how important it is to get the edges and structure just right for reliability. Testing behavior when a reminder time is already within the offset window, when an activity is paused while the user is still editing, and when the app is rebuilt from the database after returning to the list screen was the interesting work. Selecting the priority queue was simple. I added a basic idempotent send to prevent a single reminder from firing twice in a brief period of time, and I verified that the app should never schedule or send a reminder until the event is persisted. Additionally, I confirmed that past-due reminders are handled as one-time sends rather than being clamped to now on each rebuild. The purpose of applying the appropriate data structure to a user-facing feature is to make the behavior easy to understand and rationalize, and these minor adjustments achieved this goal. My practice of creating the plan first, carrying it out straight from it, and then using logs and quick tests to confirm the common path and the awkward edges the user will encounter was reinforced by the experience.