**Artifact Description**  
The artifact I selected for the algorithms and data structures category is the *Event Daddy* mobile application. I originally created this app during CS 360 to allow users to create accounts, log in, and manage a list of upcoming events. Initially, the app stored and displayed events in the order they were entered, without any kind of sorting or filtering logic.

**Justification for Inclusion**  
I included *Event Daddy* in my ePortfolio because it provided a strong opportunity to enhance how data is managed and displayed—key areas tied directly to algorithms and data structures. My improvements focused on optimizing the way the app retrieves, organizes, and displays event data. Specifically:

* **Sorting**: I implemented logic that sorts the events by date using a comparator and List.sort(). This ensures events are always shown in chronological order, rather than by insertion.
* **Filtering**: I added a real-time search bar that lets users filter events by typing part of a name or date. This required efficient string comparison and filtering logic that reacts instantly to user input.
* **Data Encapsulation**: Events are now represented using a custom Event class, making the codebase easier to manage and extend.

These updates demonstrate my ability to apply data structures (like lists of custom objects) and algorithms (like sorting and filtering logic) in a practical, user-facing application.

**Outcome Alignment**  
This work supports the following CS 499 outcome:

"Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices."

The enhancements I made clearly show thoughtful use of algorithmic thinking to improve both functionality and user experience. I believe I’ve met the outcome I set in Module One with no changes needed to my plan.

**Reflection on the Enhancement Process**  
Enhancing the app to include sorting and filtering was a valuable exercise in designing and implementing more efficient data handling. The sorting required creating a comparator that could combine and compare date and time strings correctly, while the filtering challenged me to think about performance and user experience when working with live input.

One tricky part was making sure sorting and filtering didn’t conflict. Especially ensuring that both features worked well together and didn’t break the UI. Debugging some small issues with the display and refresh logic gave me a deeper appreciation for how the front-end and logic layers connect.

Overall, these enhancements not only improved the app’s performance and usability but also helped me grow my understanding of how thoughtful data handling improves the quality of software.