Emily Walters

30 November 2025

Professor Conlan

CS499 Milestone Three Narrative

1. Briefly describe the artifact. What is it? When was it created?

   This artifact comes from the course CS360: Mobile Architecture and Programming at Southern New Hampshire University. The initial purpose of the application was to provide the user with access to an inventory with functionality to view, add, edit, and delete items. The application stores users and items in SQLite databases, and it can prompt the user via SMS when item quantities reach a certain threshold. The project began in August of 2025 and was completed for submission in September of 2025. I took this course the term prior to this Capstone course.

   This paragraph was used in a previous milestone.

2. Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in algorithms and data structure? How was the artifact improved?

   I chose to include this artifact in my ePortfolio because it is an extensive application that was coded entirely from scratch. Because of this, it showcases the complexity of the work I created, but it also allows for a significant margin of error in my work. While it is a strong display of my skills of code development both with the front end and back end, it has glaring flaws that have since been improved upon and will continue to be improved upon throughout my Capstone and beyond. There were many structural issues that needed to be addressed upon the expansion of the artifact, and it has continued to expand significantly through great simplification and greater efficiency through the implementation of Firebase as a separate platform. Implementing external tools while working within my project granted me even more experience in different areas of development.

   This paragraph is partially from the previous milestone.

3. Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

Yes, I successfully met the course outcomes I hoped to achieve with this enhancement:

- Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.
  (I will be demonstrating the usage of external tools to provide value to this project through my usage of Firebase. Using this tool will enhance and elevate the project to be more realistic of an industry-standard application through cloud storage with data hosting.)
  Through my work on this final category, I have used external tools to implement Firebase. I have elevated the project to meet better standards of how an app like this would typically function, and I employed extensive problem-solving skills and used various techniques to rehaul the software to accommodate these massive migration changes.
- Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.
  I demonstrated a security mindset that extends beyond the RBAC implementations from the initial enhancement. Through migrating to Firebase, I have allowed for the storage of data to become far more secure through not holding passwords in an insecure format, requiring authentication of users, and implementing RBAC checks through Firestore database rules specifically (they were almost too strict at first and were creating roadblocks!). I also was able to clean up missing holes from previous improvements through the reworking of the architecture that existed prior to migration.

4. Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

   Working on this last category has truly tested my patience and my skills in terms of integrating external tools into preexisting software and the changes you have to make to accommodate a migration on that scale. It was far more than just adding a couple of simple dependencies to the project and ended up relying on an entire rework of the preexisting structures to allow for Firebase storage rather than the SQLite setup from before, and this also impacted the structure of the data and the methods could not just be simply rewired to accommodate this (I ended up having

to create it as two collections: users and itemCollections, and a subcollection inside of the latter called items so that multi-user access was finally achievable).

A lot of my challenges were faced due to the massive changes to the software and avoiding null values (particularly in new values that needed to be better initialized), aligning permissions between the Firebase platform and the software, and ensuring that my error-catching was still logical with the significant changes to CRUD operations (on top of the ones I already made in category one!). Not only do the RBAC checks successfully configured in Firebase help ensure that we don't run into any more null errors outside of testing, but the extensive validation that has been added to the code will help catch some of these errors before they even exist. I now know so much more about migrating to another tool and how to follow a process that ensures functionality will persist despite the changes. This process is scary, because there will be inevitable red text and issues that arise with such massive changes to so many classes-one thing changing breaks another somewhere else and you can't get sidetracked by small errors when you have a plan for the whole overhaul. It ultimately leads to you being forced to have an even better understanding of the code you have written overall and how it all works together. I also learned how to better navigate extensive docs and apply it to writing code for something totally new (and it's a constant learning process for that!).

I also wanted to include the rules for Firestore.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

// A lot of these checks go alongside the protections in code.

    // Users collection
    match /users/{userId} {
      // Users read own profile, admins have full profile access
      allow read: if request.auth != null &&
            (request.auth.uid == userId ||

get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin');

      // Forces role to 'user' for account creation purposes-no inherent harm.
```

```
    allow create: if request.auth != null &&
          request.auth.uid == userId &&
          request.resource.data.role == 'user';

    // Users cannot update their profiles-grants a lot of power.
    allow update: if false;
    allow delete: if false;
   }

   // Item collections
   match /itemcollections/{collectionId}{
    // Users can read collections with proper permissions
    allow read: if request.auth != null &&
            (request.auth.uid in resource.data.allowedUsers ||

get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin');

    // Admin required for creating and deleting
    allow create, delete: if request.auth != null &&

get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin';

    // Collection updates is admin only for now
    allow update: if request.auth != null &&

get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin';

    // Item subcollection permissions
    match /items/{itemId}{
     // Items can be read by admin or users within list of allowedUsers
     allow read: if request.auth != null &&
             (request.auth.uid in resource.data.allowedUsers ||

get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin');

      // allowedUsers can edit items, but only admin can delete them
      allow create: if request.auth != null &&
              (request.auth.uid in request.resource.data.allowedUsers ||
```

```
get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin');

      allow update: if request.auth != null &&
            (request.auth.uid in request.resource.data.allowedUsers ||

get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin');

      allow delete: if request.auth != null &&

get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin';
    }
  }
 }
}
```