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

Joshua Brown

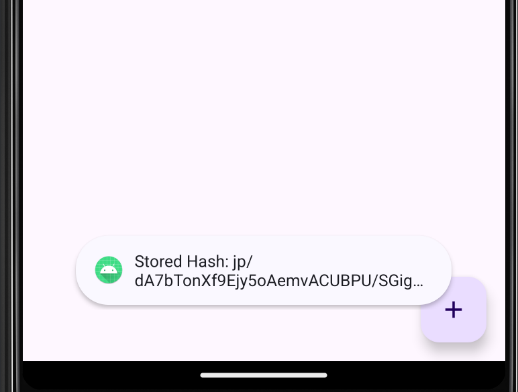
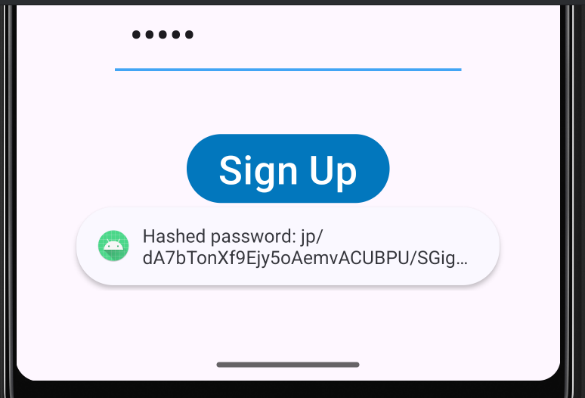
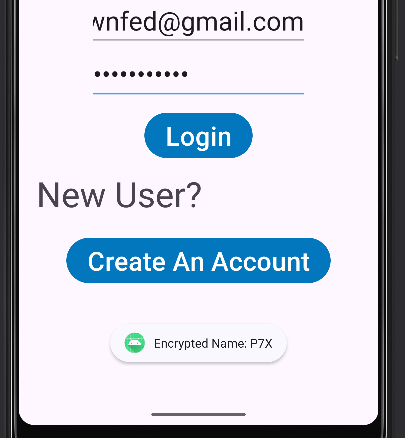
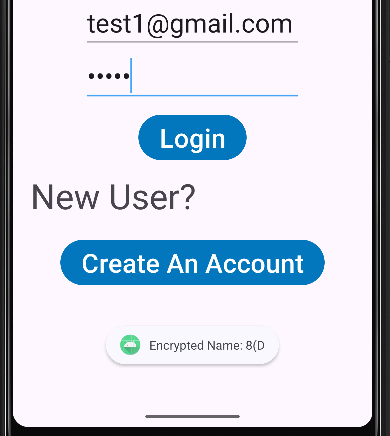
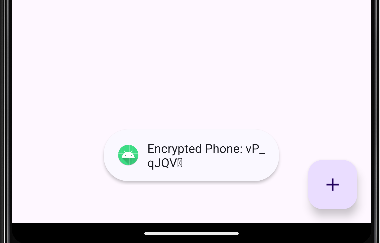
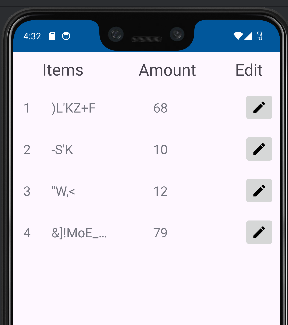
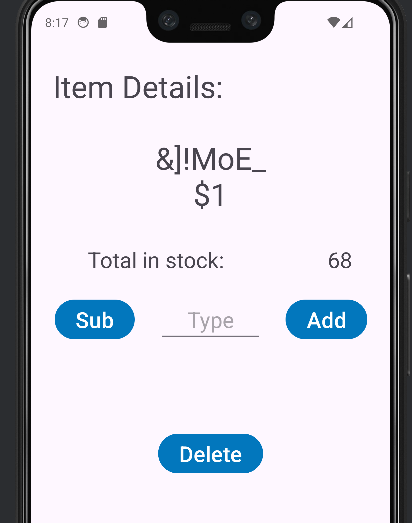
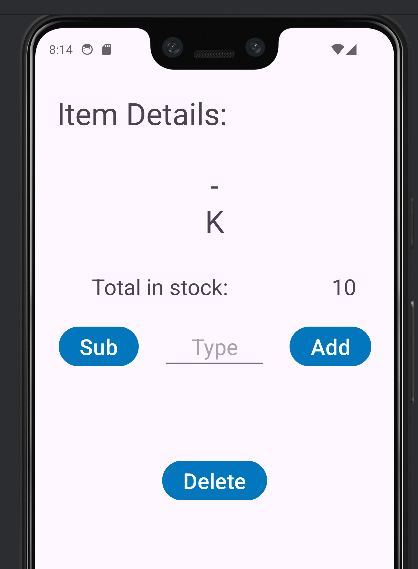
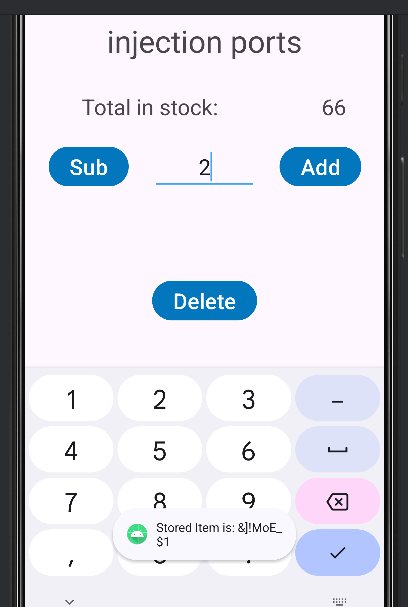
Department of Computer Science, Southern New Hampshire University

CS – 499: Computer Science Capstone

Mike Alesso, M.Sc.

November 29, 2024

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

* **Briefly describe the artifact. What is it? When was it created?**
  + The artifact for this milestone is from the final project of Mobile Architecture & Programming Class. In this project, I chose to create an inventory app that allowed a user to create and track their inventory, as well as be notified when an item is out-of-stock. The original project was created on June 30, 2024.
* **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 software development? How was the artifact improved?**
  + I chose to include this artifact, because the proper implementation of the flawed designs would not only showcase my work as a developer, but also show my dedication to a secure and functional application. For this particular Milestone, I created the algorithm to hash passwords and combined it with salt generation to develop keys for encryption. I then created algorithms to hash the user passwords upon creation, encrypt certain elements of the user data, and encrypt the items in the database. I also fixed the issue from the last Milestone where the item’s details screen was inaccessible, which was caused by the item’s ID not updating properly.
* **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?**
  + Most of the enhancements went as planned, though I struggled at first with identifying the issues from the last Milestone and coming up with a method to ensure the data would be encrypted when in the database and decrypted when shown on the screen. I had also planned to include the algorithm to update the item database when the user changes their account information or delete all items when they delete their account; however, this algorithm is tied heavily to the work that will be done in Milestone four. I did go ahead and create the functions that would be used to update all the items and delete all of the items, so they can be used once the User CRUD options have been added to the interface of the application.
  + **Screenshots showing hashed input password versus hashed stored password when logging in**
  + ****
  + **Screenshots of showing user’s name is encrypted in the database**
  + ****
  + **Screenshot showing user’s phone is encrypted in the database if the user has one stored.**
  + ****
  + **Screenshot showing encrypted items from database without decrypting them**
  + ****
  + **Screenshots showcasing that after data is decrypted to the screen, it is encrypted again when in transit.**
  + ****
  + **Screenshot showing item data is encrypted when stored in the database.**
  + ****
* **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?**
  + **Creation of a Password Hasher Class**
    - Removed salt generation function from User Class and added it here
    - Made generate salt function return a string, so it can be stored in the User table
    - Included a decode salt function, so the hashed password could be combined with the hash bytes to create a key.
    - Created a function to hash a password that combined the salt bytes from the decode salt function to create a key.
    - Created a function to Compute Hash that is combined with Validate Hash to test if data is already encrypted. I had an issue when utilizing the XOR encryption algorithm, because it can be used for encryption and decryption, I sometimes ended up encrypting when I wanted to decrypt and vice versa.
    - Created an XOR function to encrypt data using a key, which is utilized by my encrypt and decrypt functions, which were created to help prevent confusion on which function I wanted to use.
    - Thoroughly commented code.
  + **User Repository Update**
    - Refactored create method to generate salt at the creation of a user and hash the provided password. The new hashed password would be combined with salt to create a key and stored as the new password in the database along with a string version of the salt, which would be used to hash the user password when they attempt to login for comparison.
    - Refactored update method to encrypt the name and phone number data, on update. Did not include email, password, and salt to prevent errors in the code.
    - Thoroughly commented code
  + **New User Activity** 
    - Since I only included the encryption portion of the user repository in the update method. I included the refactored create method to create the user, which creates the salt, hashes the password, and creates the key, then I included the update method to encrypt the other data.
    - Thoroughly commented code.
  + **Login Activity Update**
    - If an email is found in the database from the provided input, then the string version of the salt is pulled from the user database and converted to bytes using the decode salt function, then password input is hashed and combined with the salt bytes to create a key. If this key matches the password in the database, then it pushes the user to the next screen. If not, then it prompts the user that the provided password was incorrect.
    - Thoroughly commented code
  + **Inventory Repository Update**
    - Refactored Create method to encrypt data after creation to store in the database.
    - Read One method was not operating correctly and was causing problems in the last Milestone. Updated method to handle errors and made sure the method included the item ID properly.
    - Refactored the Read All method to decrypt the data after pulling it from the database. Also, included a reference indicator to keep track of which numbered item was next in the database and update the item ID if the ID did not match the expected reference indicator due to an earlier deletion from the database.
    - Refactored the update method to encrypt an item’s data upon update to the database
    - Included Update Correction ID method to be utilized with the Read All method. This method is called to correct an item’s ID by looking up the previous Item ID and replacing it with the correction.
    - Included Update All method, which will be utilized to update all inventory items for a user in the event that the user changes their username or password. Utilizes subfunctions Update Correction Email and Update Correction Password
    - Included Update Correction Email, which takes an email as the correction and updates the email in the database to the correction.
    - Included Update Correction Password, which takes a password as the correction and decrypts the item using the old password, and re-encrypts the item using the new password, updating it in the database.
    - Included Delete All method, which deletes all items from a particular user when they delete their account.
    - Commented code thoroughly.
  + **Inventory Activity Update**
    - Fixed issues where items were not being passed correctly to Item Detail Activity.
    - Included an item ID in the recycler view for an item, so the item’s ID is visibly displayed with the listed item.
  + **Item Detail Update**
    - Fixed issues that prevent screen from not displaying and updating information.