Development of a Biometric File Encryption Mobile Application

`SOFE 4640U - Mobile Application Development - Group 4
Okiki Ojo - 100790236
Ahmed Darwish - 100754743
Daniel Amasowomwan - 100787640
John Howe - 100785128
Johvonne Keane - 100784273

GitHub Repos: okikio-school/vault & okikio-school/vault-cloud

Project Details and Results

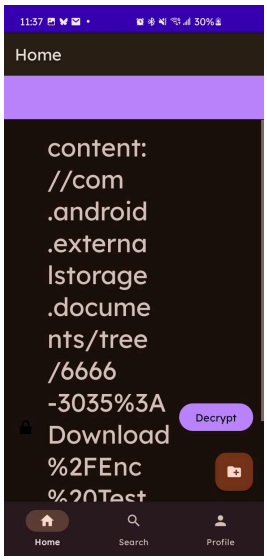
Project Description

There are many computer systems which store information that can be sensitive or confidential. This information needs to be protected from unauthorized access. Traditionally, securing files is done using passwords and PINs, but these procedures are vulnerable to weak passwords, phishing attacks, and social engineering. Additionally, passwords can be easily forgotten, lost, or stolen. A biometric solution can offer the security needed for sensitive files, without the existing weaknesses of passwords. Biometric technologies like fingerprint recognition and facial recognition have become a common feature in mobile devices, so there is an opportunity to make this encryption method accessible to everyday users. Our application takes advantage of these biometric technologies to encrypt files locally. We have chosen to name the application “Vault” and we will be referring to the encrypted folders as “vaults” from this point forward.

Project Results

Our project is divided into a few pages: the home screen, search screen, file picker screen, profile screen, and intro screen. These screens each provide features to the user, each of which we will showcase in this report.

Home Screen

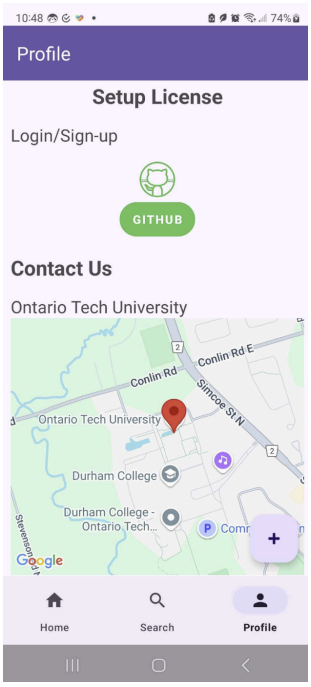


The home screen contains a list of all the users' vaults. Each vault includes the vault name, the vault directory, and a “Decrypt” button, which allows the user to decrypt the vault. We used an SQLite database and a DatabaseHelper class to store and retrieve vault information. The database itself is not encrypted, but the data is stored as an encrypted string in one of the table fields.

Search Screen

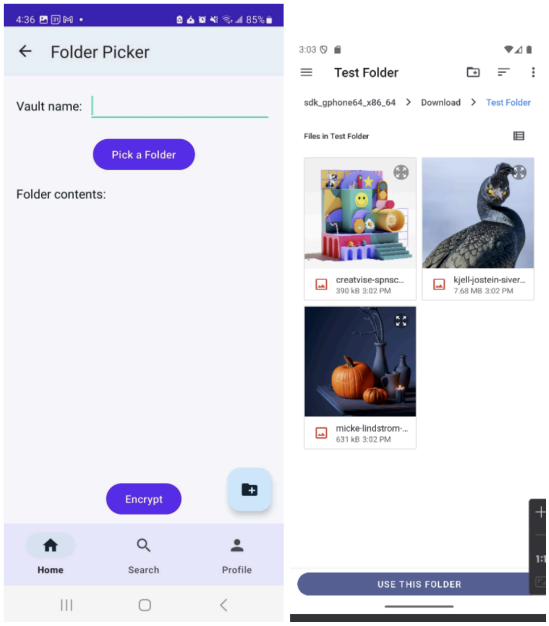
The search screen contains a list of vaults like the home screen, but this screen includes a search bar, which can be used by the user to search for vaults by title. The list is updated as everytime text in the search bar is changed.

Profile Screen



In the profile screen, users can log in to their account using GitHub OAuth. OAuth is handled on a cloud server which returns a confirmation message on a cloud-hosted website using Vercel. The user database is also handled in the cloud. The profile screen also contains a “Contact Us” section with an integrated Google Maps component that contains a marker at Ontario Tech University.

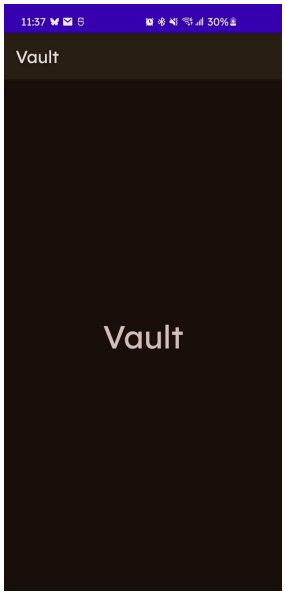
Folder Picker (New Vault) Screen



Each of the previously described screens includes a floating action button with a “plus” sign that is used for adding new vaults. The button redirects the user to this screen, which provides a text input for entering

the vault title. The screen also includes a button to open a folder picker activity, which allows the user to select a folder from their device. The contents of the folder are displayed in text below the button, and finally a button at the bottom of the screen allows users to encrypt the folder and store the vault information and encrypted data on a local database.

Intro Screen



The intro screen is where the user can play an introductory video to get a quick introduction into the app. This video is only loaded upon the first time of opening the app, and if they'd wish to see the video again they must do so with a fresh installation of the app.

Implementation Challenges

During the development of the Vault mobile application, we encountered a few challenges. One of the primary issues we faced was SDK synchronization. Ensuring that all team members were using the same SDK versions and dependencies was difficult to do at first and caused some git conflicts and integration issues during development. Additionally, implementing OAuth for our application was quite difficult. Connecting our cloud server to the application required a lot of troubleshooting and refactoring of our existing code to make the system work properly. Another major hurdle in the project was integrating the libsodium library with our project. Finally, implementing and demoing the application on real Android devices introduced additional challenges. For example, although we developed our application for SDK 35 and tested our system on SDK 35 AVMs, we had to consider how certain features, like our app bar, worked in older versions like Tiramisu. This issue and others caused runtime errors that set back the program, especially closer to the presentation date.

Lessons Learned

This project has taught our team many crucial lessons that have changed our perspective on how to pursue team projects from now on. To begin, being able to communicate clearly with your team members in order to match the deadline/schedule is important. Without communication, there would be no way for us to proceed and complete the desired project. In addition, preparation and dividing the workload was an essential component in the success of this project. The group must ensure that the work was divided evenly and equally across all members for a successful outcome. This project taught us a lot in regards to mobile development and how to deploy a complete project on android devices. It familiarized us with Kotlin as a language, android studios as a software, and how a project can be implemented on an actual device and deployed.