**Final Project Report**

**for**

**Social Engineering Training Game**

**Version 1.0**

**Prepared by: Frederic Stresau, Simon Chummar, Alexander C. Bodoh,   
Quy Pham, Jackson Harper**

**Florida Polytechnic University SSE Group 7**

**4/24/2024**

**1.2) Abstract.**

The goal of this project is to design and develop a game (software) to evaluate user's awareness about social engineering attacks in an entertaining game format. It is a simple mobile game application that asks users questions about different types of social engineering attacks and basic protection techniques, provides feedback on user answers, and displays correct answers. The game application allows creation of separate accounts for each user, with each account’s data protected from external access through encryption. A user profile page is also available to show users their level of improvement over time.

**1.3) Core Design.**

**1.3.1) System Architecture.**

For the system architecture, there was no database management system since everything was stored locally. All the data such as past scores, usernames, passwords, and questions are stored in encrypted JSON files. The passwords were also hashed using SHA-256 and the files were encrypted using MD5 and DES. We utilized the Repository system architectural style, where main game components interact with a central repository which takes the form of the JSON files (both encrypted score history and unencrypted questions).

The application was developed in Android Studio and used the Java programming language. We used the libraries Moshi and Jasypt. Moshi is a Java software library for parsing and encoding JSON data and Jasypt is a library for encrypting and decrypting. Android Studio also includes Android device emulators which makes the testing process much easier.

**1.3.2) System Features.**

Core features of the system are as follows:

* Account creation
  + Use case: A new user creates an account by providing a username and a secure password.
* Log-in system
  + Use case: An existing user logs into the system by providing their previous username and password.
* Enforcement of strong password requirements.
  + Use case: A user attempts to create an account and provides an insecure password, and is thus prevented from creating an account.
* SHA-256 password hashing to prevent attackers from learning user passwords.
  + Use case: A user creates an account in the system, and their password hash is saved instead of the plaintext user password.
* Encryption of user score history through MD5 and DES, with the salted password SHA-256 hash used as the encryption key.
  + Use case: An existing user has a profile, and their password hash is visible on the disk, but their profile data is not decipherable by attackers since the salted password is not known.
* A five-question quiz game with four potential answers per question (one correct), where questions are randomly selected from a pool of 36 questions about social engineering and cybersecurity.
  + Use case: An existing user plays a round of the game and selects their answers to each question.
* Feedback on correct answers for each question.
  + Use case: A user answers a question incorrectly and receives feedback on what the correct answer is.
* Profile page displaying previous scores in reverse chronological order.
  + Use case: A user finishes a game round and checks their previous score history to see how they’ve improved over time.

**1.3.3) Screenshots.**

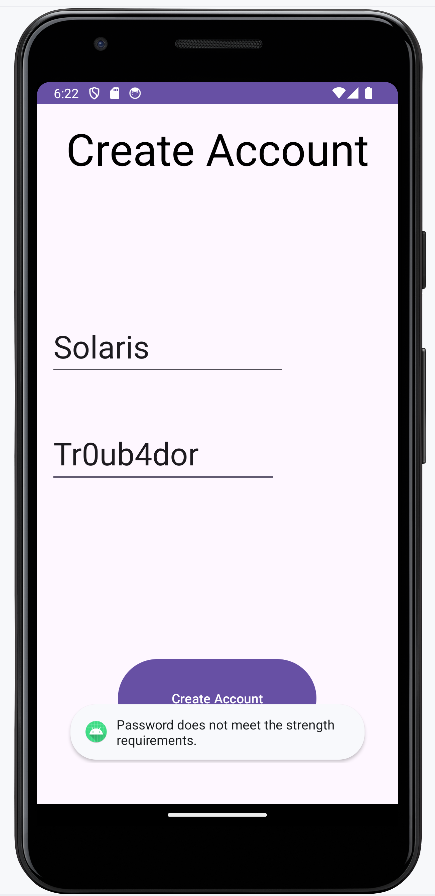
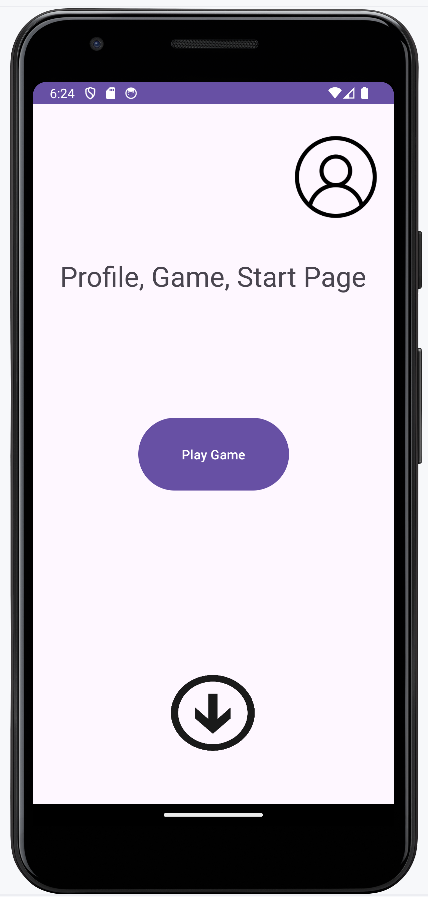
**  **

Figure 1.3.3.1) Screenshots: a) Account creation with invalid password; b) Logging in; c) Home screen

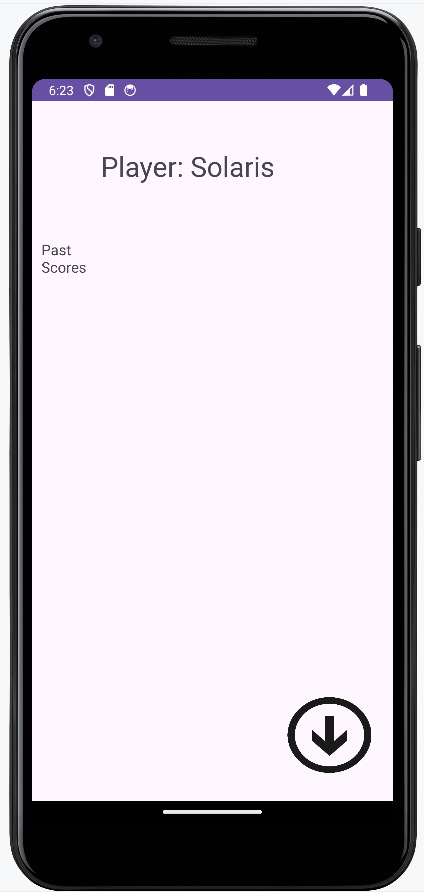
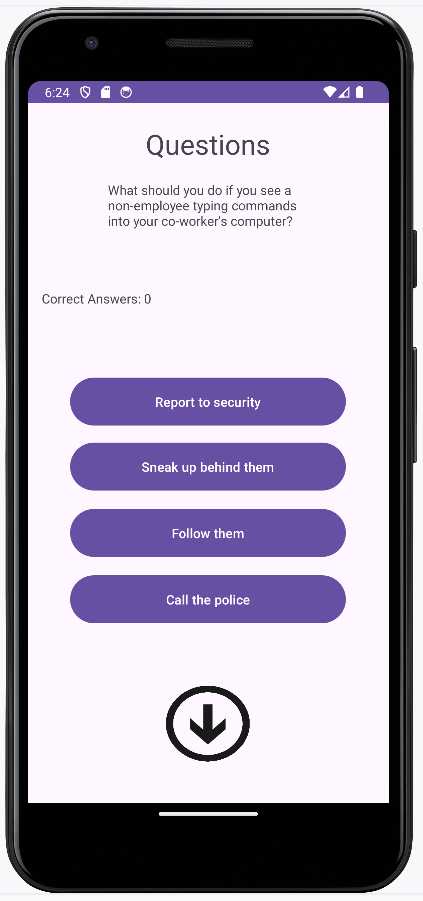
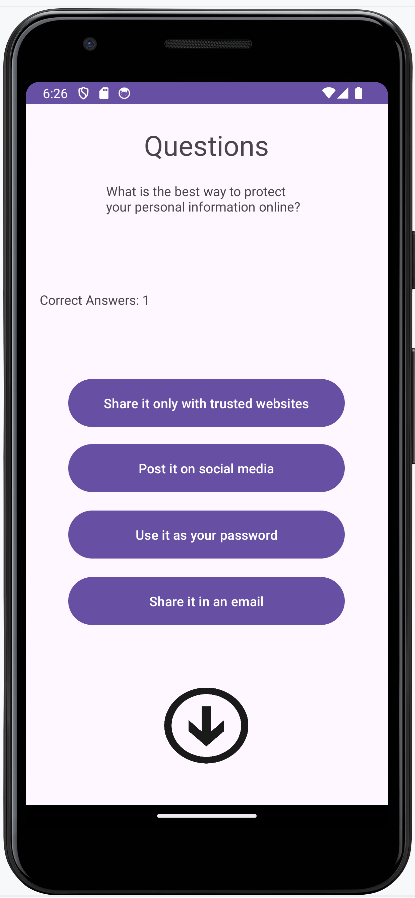
**  **

Figure 1.3.3.2) Screenshots: a) profile page for new account; b) Question screen; c) another question example

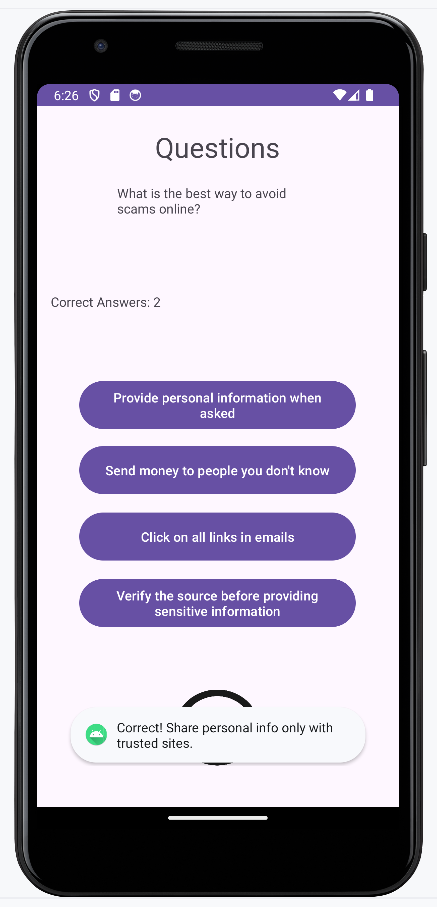
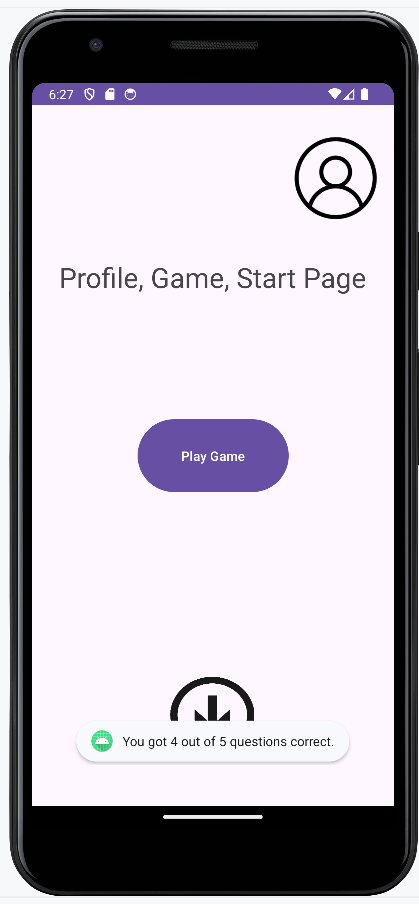
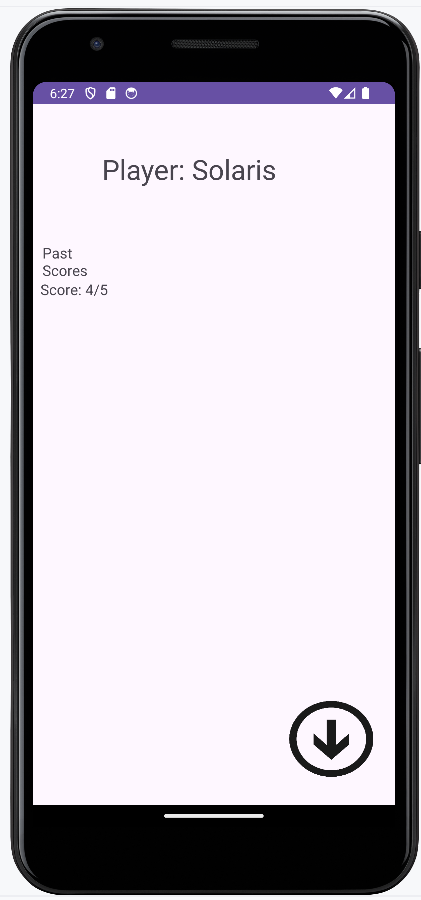
**  **

Figure 1.3.3.3) Screenshots: a) feedback on correct answer; b) quiz game results; c) profile page showing a previous score

**1.3.4) Source Code.**

[alexander-c-b/secure-software-engineering (github.com)](https://github.com/alexander-c-b/secure-software-engineering)

**1.3.5) Video Demo.**

[Team 7 Final Project Demo Video - Made with Clipchamp\_1713928797886.mp4 - Google Drive](https://drive.google.com/file/d/1CTOLaWfjc9Owk1hBvOxS_aXejiXRwz-p/view)

**1.4) Testing Results.**

**1.4.1) Unit Test Results**

* Encryptor
  + ✓ Encrypts/decrypts file given salted hash
  + ✓ Salting is used so that the filename hash is different from the hash for encryption
* JsonEncoder
  + ✓ Loads question data from JSON file
  + ✓ Decrypts then loads JSON score history
  + ✓ Encrypts then saves JSON score history
* GameFile
  + ✓ Reads string from filename in app’s private data directory
  + ✓ Writes string to filename in app’s private data directory
* Authenticator
  + Account Creation
    - ✓ Invalid non-alphanumeric usernames prohibited
    - ✓ Password strength requirements enforced
      * 12+ characters
      * Capital, lowercase, number, special character
    - ✓ User is informed of issues or successful account creation
  + Login
    - ✓ Existing users can log into system
    - ✓ Invalid combinations of username and password cannot proceed
    - ✓ Raw passwords are not stored
* GameRoundControl
  + ✓ Game will end after five questions
  + ✓ Checks for correct answer & gives feedback
* GameContent
  + ✓ Loads questions from JSON file
  + ✓ Randomizes questions
  + ✓ Does not repeat questions in the same round
* Question
  + ✓ Creates question objects correctly
  + ✓ Ensures that correct answer index aligns with correct answer choice
  + ✓ Ensures correct answer string is retrieved
* GameUI
  + ✓ Displays questions
  + ✓ Displays answer choices
  + ✓ Displays answer feedback
* User
  + ✓ Stores username
  + ✓ Stores hashed password
  + ✓ Stores salted hashed password
* Profile UI
  + ✓ Displays username
  + ✓ Displays past scores

**1.4.2) Integration Testing Results**

* ✓ Create Account creates an encrypted score file
* ✓ Logging in provides access to the profile and game
* ✓ Playing a game uses questions loaded from the on-device question bank
* ✓ Scores from previous games are visible in profile

**1.4.3) System Testing**

* ✓ Performance Testing: Using the Android Studio device emulator, each function of the app was tested for performance, and all of the UI elements respond within 2 seconds.
* ✓ Functional Testing: The end-to-end use-case is fully functional, as are sub-functions within it:
  + User creates account
  + Logs in
  + Plays game
  + Views previous score
  + Logs out
* ✓ Security testing: Attempts to access user data via unauthorized means, such as viewing device files directly instead of through the app, are not able to read the encrypted score history. Furthermore, user scores cannot be recovered even when the attacker knows the unsalted password hash which is stored in the private data directory.

**1.5) Conclusions.**

We were able to create a social engineering training game which has the feature set necessary for multiple users to independently play the game, learn from their correct answers and mistakes, and see their history, while being unable to gain unauthorized access to other users’ data. Our system architecture relied on simplicity of implementations as well as high-quality libraries to enhance the maintainability of the software.